

LJ-83-19

SOUTHWEST FISHERIES CENTER

P.O. BOX 271
LA JOLLA, CA 92038

NATIONAL MARINE FISHERIES SERVICE

SOUTHWEST FISHERIES CENTER

AUGUST 1983

DOCUMENTATION OF TOOTH- READING RECORDS AND AGE-DATA FILES FOR THE NORTHERN OFFSHORE SPOTTED DOLPHIN

by

Aleta A. Hohn and Rebecca J. Hankins

Library
NOAA, National Marine Fisheries Service
SWFC Honolulu Laboratory F/SWC2
2570 Dole Street
Honolulu, HI 96822-2396

ADMINISTRATIVE REPORT LJ-83-19

"This report is used to insure prompt dissemination of preliminary results, interim reports, and special studies to the scientific community. The material is not ready for formal publication since the paper may later be published in a modified form to include more recent information or research results. Abstracting, citing, or reproduction of this information is not allowed. Contact author if additional information is required."

DOCUMENTATION OF TOOTH-READING RECORDS AND
AGE-DATA FILES FOR THE NORTHERN OFFSHORE SPOTTED DOLPHIN

by

Aleta A. Hohn and Rebecca J. Hankins

Southwest Fisheries Center
National Marine Fisheries Service, NOAA
La Jolla, California 92038

August 1983

ADMINISTRATIVE REPORT NO. LJ-83-19

CONTENTS

	Page
BACKGROUND AND PURPOSE.....	1
TOOTH-READING-RECORD CODE FORM AND USAGE NOTES.....	2
EDITING THE TOOTH-READING-RECORD DATA BASE.....	2
TOOTH-READING EDIT PROGRAM.....	5
SPECIMEN AND DATA ERRORS.....	5
TRSPTEDT CHANGES FOR OTHER DOLPHIN STOCKS.....	10
NAMING CONVENTION FOR THE INPUT AND OUTPUT FILES.....	12
DATA EXTRACTION PROGRAMS.....	13
ACKNOWLEDGEMENTS.....	19
LITERATURE CITED.....	20
APPENDICES	
Appendix 1: Tooth-Reading-Record Usage Notes.....	21
Appendix 2: VAX Fortran Northern Offshore Spotted Dolphin Edit Program.....	30
Appendix 3: User's Guide to the Tooth-Reading Edit Process.....	41
Appendix 4: VAX Fortran Data-File Format Programs	45
4A GOODBAD.....	46
4B LASTDATE.....	47
4C REFORMAT.....	48
4D INDEX.....	50
4E MALE.....	51
4F MFSPLIT.....	52
Appendix 5: VAX Fortran Data-Extraction Programs.....	53
5A MAGEBST.....	54
5B MAKEDAT.....	58
5C MAKEDAT2.....	60
5D OUTLIER.....	67
Appendix 6: Age-Data File Formats.....	68
6A XXTRFSPT.....	69
6B TRFSPTDB.....	70
6C AGEBESTF.....	70
6D FLHAGE.....	71
6E FSETAGE.....	71

LIST OF TABLES

Table		Page
1	Variable verification cross-reference.....	6
2	Spotted dolphin tooth-reading edit logical- error checks.....	8
3	Output for one logical group from the Tooth- Reading Record edit program (TRSPTEDT).....	11

LIST OF FIGURES

Figure		Page
1	Tooth-Reading-Record Code Form.....	3
2	Spotted dolphin data files flowchart showing the steps used to generate the age data base. Although this chart illustrates the female data files, the procedure is the same for the male data, but the "F" would be replaced by an "M" in all data file names. The symbols are the same as in Figure 3.....	14
3	Spotted dolphin age data files detailed flowchart showing data extraction programs and input and output data files. Although this chart illustrates the female data files, the procedure is the same for the male files.....	15
4	Flowchart showing data retrieve domains available for the age data and related data bases. Symbols are the same as in Figure 3.....	16



Digitized by the Internet Archive
in 2025

DOCUMENTATION OF TOOTH-READING RECORDS AND
AGE-DATA FILES FOR THE NORTHERN OFFSHORE SPOTTED DOLPHIN

Aleta A. Hohn and Rebecca J. Hankins

Southwest Fisheries Center
National Marine Fisheries Service, NOAA
La Jolla, California 92038

BACKGROUND AND PURPOSE

The northern-offshore-spotted-dolphin (Stenella attenuata) age-determination project began in the fall of 1980 with the purpose of obtaining ages for a large sample of specimens. Although a smaller sample from this stock of dolphins had been aged in a previous study (Perrin, Coe, and Zweifel, 1976), the number of available specimens has increased nearly five-fold and includes data spanning 12 years. In addition, cetacean age-determination techniques have been refined. Age estimates are assumed to be more accurate with current techniques because the resolution of growth layers groups (GLGs, terminology of Perrin and Myrick, ed., 1980) is better, and the interpretation of GLGs is thought to be more accurate. The large number of age readings made in this sample required that a data base be created and that a system for editing the data be established.

Two samples were used for the age-determination project. The initial sample was obtained from randomly-generated listings, by sex, of the entire sample of northern offshore spotted dolphins collected from 1973 through 1978 (approximately 7,850 records). The first 800 specimens of each sex, from which teeth were available, comprised the sample. Two independent readers made at least three readings (age estimates) for each of the 1,600 specimens. The second sample comprised all of the specimens of female northern offshore spotted dolphin with teeth collected during 1981 (n=315). One age estimate was made by each of the two readers. For both samples, readings were made independently by the two readers and without knowledge of the previous readings or associated biological data. The procedures used to prepare and "read" the teeth are described in Myrick et al. (1983). Each reading was coded onto a form, the Tooth Reading Record, and entered into a data file which was to become the Tooth Reading Record or "age" Data Base.

The purposes of this report are to document the creation of the northern-offshore-spotted-dolphin age-data base for users of the age data and to standardize the procedures to create age-data bases for other species or stocks of dolphins. This documentation includes the development and use of coded Tooth Reading Records, the procedure for editing the age-data base, and a description of the programs written for that purpose. In addition, it provides descriptions and listings of the programs used for data extraction

from other dolphin-data bases, such as life-history and set-log data bases, for age-related biological analyses. The same record format and edit procedure for the northern-offshore-spotted-dolphin age-data bases is expected to be used to create age-data bases for other dolphin species, such as the spinner dolphin, S. longirostris.

TOOTH-READING-RECORD CODE FORM AND USAGE NOTES

The Tooth-Reading-Record (TRR) code form (Figure 1) was developed to standardize the description and recording of GLG counts and characteristics. It was designed to document the preparation, quality and reading of tooth sections used for age determination. The Tooth Reading Record comprises 38 elements divided into eight sections: Specimen General, Condition of Slide, Reading General, Age Recorded, Dentinal, Cemental, Code Tables, and Notes.

Seven of the elements on the Tooth Reading Record may not be left blank. Four of these are marked by an asterisk on the Tooth Reading Record. These are Specimen Number, Reader Code, Number of Dentinal GLGs, and Number of Cemental GLGs. If best, high and low dentinal or cemental GLG counts are not made, or if an Age Recorded (not yet marked with an asterisk) is not estimated, these elements must be filled in with the missing-value code, 999, to prevent any blanks from being read as zero, which is a valid age estimate. The two additional elements not to be left blank, yet not marked with an asterisk on the code form, are Reading of Day and Reading Date.

The Usage Notes (Appendix 1) for the Tooth Reading Record define and explain each element on the code form and give the coded options used to describe an element. These options are defined as explicitly as possible, to minimize differences in interpretation among readers. For instance, the six "Condition of Slide" elements are defined in Code Table 4 of the Usage Notes and can be used to determine whether a slide of tooth thin sections meets the criteria necessary to code for a positive answer to each element. There are ten code tables for the ten elements that require a coded response on the Tooth Reading Record, and each table contains the possible responses for those elements.

The Tooth Reading Record Code form was not created until after some of the age-determination readings had been done. The elements included on the coding form are considered to be the most important for interpreting the quality of a reading based on the readers' experience. (Since some of the readings were made before the code forms were available, these were transcribed to the forms using notes taken by the readers.)

EDITING THE TOOTH-READING-RECORD DATA BASE

The Tooth-Reading-Record data base comprises all the tooth readings made and recorded on the Tooth-Reading Records. The records contained in the data base were edited using a combination of FORTRAN language computer programs written specifically for the tooth-reading code form, subroutines written for

SPECIMEN, GENERAL

Figure 1. The Tooth-Reading-Record data code form.

editing other dolphin data bases, and by visual examination of the data. The last procedure is important to assure, for example, that specimen preparation data are consistent from reader to reader and that the counts made for a specimen are all valid readings (i.e., not designated as "recuts" or "remounts"). The following summarizes the procedures for editing the data base.

1. Tooth Reading Record code forms are keypunched and then transferred to disk on the VAX computer system.
2. The raw data file is sorted by Reader Code and Specimen Number [using the command: \$SORT/KEY=(POS:30,SIZE:2)/KEY= (POS:4,SIZE:7) input-file-name output-file-name]. Data are then manually checked to insure that the required number of tooth readings were made for each specimen. Three valid counts per specimen per reader were required for part of this project. New counts are added, if needed, by appending them to the original data file.
3. The completed data file is sorted by Reader Code, Reading Date, and Reading-of-Day [\$SORT/KEY=(POS:30,SIZE:2)/KEY= (POS:33,SIZE:6)/KEY=(POS:30,SIZE:3) input-file-name output-file-name] before running the tooth-reading edit program, TRSPTEDT. Errors flagged by the edit program are written into TRSPTEDT.LOG, and must be evaluated manually. Any necessary corrections are then made to the data base and the original Tooth Reading Record forms.
4. Concurrently with number 3, readers check each count with previous counts for a specimen to detect "mix-ups" of teeth which may occur when samples are collected or when slides are prepared. If a preparation is found to have been labeled with the wrong specimen number, a specimen "mix-up" has occurred (see "Specimen and Data Errors," below), and a new tooth for that specimen is prepared and read. The new reading is then added to the data base.
5. When a specimen "mix-up" occurs, the mislabeled slide is "bad" and the data record on disk is marked with an asterisk in column 80 with the VAX editor. This identifies the reading so it can be separated from the correct records (see GOODBAD). Simultaneously, the original Tooth Reading Record code form and the prepared slide are prominently labeled as "bad".
6. When all readings are complete, the data file is edited again until all errors have been corrected. An additional data check is made by using simple histograms and scatterplots (e.g., length on age, corpora on age, color pattern on age) to identify outlying data points. After the edit procedure has been completed, the current version of the data file, XXTRFSPT.ALL, is stored on magnetic tape with all user "help" files, documentation, and the DATATRIEVE dictionary. This process is handled by the data manager. Additional errors may be detected as analyses are performed, requiring that the master data base be updated. Corrections to any data base under control of the data manager are made by the data manager after the corrections are brought to his/her attention.

Tooth-Reading Edit Program

The tooth reading edit program, TRSPTEDT (Appendix 2), was designed to identify possible errors in the data and to print a report indicating the types of errors encountered. The program contains range and logical-error checks written specifically for spotted-dolphin age data. These checks identify "errors" as data that do not conform to the criteria specified. For example, the range check for specimen number requires that three alphabetic characters (the observer's initials) are located in columns 4-6. If the columns contain blanks or numerics, the record will be flagged by the program and printed out. Some "flagged" records may remain in the data base because all flags do not necessarily represent errors. Flags identify records which fall outside the specifications of the edit program, e.g., when an observer has only two initials. Range checks used for the spotted dolphin age data are listed in Table 1; logical-error checks are in Table 2.

The output from the tooth-reading edit program is designed to make error identification and correction as efficient and easy as possible. Initially, the Tooth-Reading-Record data base is read by a pre-edit subroutine (CS00TR) to insure that the records are sequenced properly before the main edit program is run. That is, the data base must first be sorted by Reader Code, Reading Date, and Reading of Day, the unique key for each record. If any records are out of sequence or duplicate records are found, these are printed out and the edit program stops. Once the sequence errors have been corrected, the edit program can be resubmitted to perform the range and logical-error checks. Errors are printed out in sequence with the error-free records of a "logical group". A logical group contains from 1 to 10 records with the same Reader Code and Reading Date (Table 3). If a logical group contains no errors, it is not printed. The edit program, TRSPTEDT, must be run iteratively until all reconcilable errors have been corrected. Specific information on the edit program is available in the program documentation and in the "User's Guide for the Tooth-Reading-Record Data-Base Edit Process" (Appendix 3).

Specimen and Data Errors

In addition to the range and logical errors which the edit program identifies, errors in specimen-number assignment may occur. These errors must also be found and corrected. Three types of specimen-number errors were defined for the northern-offshore-spotted-dolphin age sample. First, the data for the specimen were incompatible with the age of the specimen as a result of a specimen number "mix-up." Second, teeth from different animals inadvertently had been assigned the same specimen number sometime after the teeth had been removed from the jaw sections. When it was not possible to determine which tooth was correct, by comparing the age estimates to the life-history data, the specimen was removed from the sample. Third, one of the two situations above occurred but could not be resolved because there were no more teeth available for that specimen. Seventeen specimens from the female sample fell into one of these categories, but none of the specimens from the male sample were affected. In addition, six of the specimens (three males and

Table 1 - VARIABLE VERIFICATION CROSS-REFERENCE

VARIABLE NAME	INTERNAL FIELD EDITS					OTHER VERIFICATIONS
	CARD COL	CHAR TYPE	BLANK OK?	RANGE		
				LOWER	UPPER	
READING OF DAY	1-3	N	NO	1	1	MUST BE FILLED FOR SORTING.
SPECIMEN NUMBER	4-6	A	NO	AAA	ZZZ	
	7-10	N	NO	1	999	
TOOTH OF SPECIMEN	11	N	YES	1	9	
SLIDE OF TOOTH	12	N	YES	1	9	
PREPARATION DATE:						
Year	13-14	N	YES	79	83	*
Month	15-16	N	YES	00	12	
Day	17-18	N	YES	00	31	
PREPARATOR	19-20	N	YES	1	6	
TOOTH TREATMENT	21	N	YES	1	2	
MOUNTING MEDIUM	22	N	YES	1	3	
FADED	23	N	YES	0	1	
SHRINKAGE	24	N	YES	0	1	
OFF-CENTER	25	N	YES	0	1	
DARK	26	N	YES	0	1	
LIGHT	27	N	YES	0	1	
SHREDDED	28	N	YES	0	1	
UNKNOWN	29	N	YES	0	1	
READER CODE	30-31	N	NO	6	18	*
REQUEST REMOUNT/RECUT	32	N	YES	1	3	
READING DATE: Year	33-34	N	NO	01	83	* { IF NO READING DATE, NEED INTEGER DUMMY CODE FOR SORTING.
Month	35-36	N	NO	01	12	
Day	37-38	N	NO	01	31	
MICROSCOPE I.D.	39-40	N	YES	1	2	
PULP CAVITY CONDITION	41	N	YES	1	3	
AGE RECORDED: Best	42-44	N	YES	0	999	
High	45-47	N	NO	0	999	
Low	48-50	N	NO	0	999	
DENTINAL GLG CONDITION:						

A=ALPHA B=BLANK C=COMMA N=NUMERIC

[illegible]

*Range will change for spinners or other stocks.

**Table 2 - SPOTTED DOLPHIN TOOTH READING EDIT LOGICAL
ERROR CHECKS**

ERROR001	IF THE PRESENT OCCURRENCE OF READER CODE (1.30-1.31) IS NON BLANK ... THE PRESENT OCCURRENCE OF READER CODE SHOULD BE EQUAL TO THE PREVIOUS OCCURRENCE OF READER CODE.
ERROR002	IF THE PRESENT OCCURRENCE OF READING DATE (1.33-1.38) IS IDENTICAL TO THE PREVIOUS OCCURRENCE OF READING DATE ... THE PRESENT OCCURRENCE OF READING OF DAY (1.01-1.03) SHOULD BE EQUAL TO THE PREVIOUS OCCURRENCE OF READING OF DAY PLUS ONE.
ERROR003	IF READING DATE (1.33-1.38) IS NON BLANK AND PREPARATION DATE (1.13-1.18) IS NON BLANK ... READING DATE MUST BE GREATER THAN PREPARATION DATE.
ERROR004	IF AGE RECORDED BEST (1.42-1.44) IS NOT EQUAL TO 999 AND AGE RECORDED HIGH (1.45-1.47) IS NOT EQUAL TO 999 ... AGE RECORDED HIGH MUST BE GREATER THAN OR EQUAL TO AGE RECORDED BEST.
ERROR005	IF AGE RECORDED BEST (1.42-1.44) IS NOT EQUAL TO 999 AND AGE RECORDED LOW (1.48-1.50) IS NOT EQUAL TO 999 ... AGE RECORDED LOW MUST BE LESS THAN OR EQUAL TO AGE RECORDED BEST.
ERROR006	IF AGE RECORDED HIGH (1.45-1.47) IS NOT EQUAL TO 999 AND AGE RECORDED LOW (1.48-1.50) IS NOT EQUAL TO 999 ... AGE RECORDED HIGH MUST BE GREATER THAN AGE RECORDED LOW.
ERROR007	IF DENTINAL GLGS BEST (1.54-1.56) IS NOT EQUAL TO 999 AND DENTINAL GLGS HIGH (1.57-59) IS NOT EQUAL TO 999 ... DENTINAL GLGS HIGH MUST BE GREATER THAN OR EQUAL TO DENTINAL GLGS BEST.
ERROR008	IF DENTINAL GLGS BEST (1.54-1.56) IS NOT EQUAL TO 999 AND DENTINAL GLGS LOW (1.60-1.62) IS NOT EQUAL TO 999 ... DENTINAL GLGS LOW MUST BE LESS THAN OR EQUAL TO DENTINAL GLGS BEST.
ERROR009	IF DENTINAL GLGS HIGH (1.57-1.59) IS NOT EQUAL TO 999 AND DENTINAL GLGS LOW (1.60-1.62) IS NOT EQUAL TO 999 ... DENTINAL GLGS HIGH MUST BE GREATER THAN DENTINAL GLGS LOW.
ERROR010	IF CEMENTAL GLGS BEST (1.67-1.69) IS NOT EQUAL TO 999 AND CEMENTAL GLGS HIGH (1.70-1.72) IS NOT EQUAL TO 999 ... CEMENTAL GLGS HIGH MUST BE GREATER THAN OR EQUAL TO CEMENTAL GLGS BEST.
ERROR011	IF CEMENTAL GLGS BEST (1.67-1.69) IS NOT EQUAL TO 999 AND CEMENTAL GLGS LOW (1.73-1.75) IS NOT EQUAL TO 999 ... CEMENTAL GLGS LOW MUST BE LESS THAN OR EQUAL TO CEMENTAL GLGS BEST.

Table 2 (Continued)

ERROR012	IF CEMENTAL GLGS HIGH (1.70-1.72) IS NOT EQUAL TO 999 AND CEMENTAL GLGS LOW (1.73-1.75) IS NOT EQUAL TO 999 ... CEMENTAL GLGS HIGH MUST BE GREATER THAN CEMENTAL GLGS LOW.
ERROR013	IF DENTINAL GLGS BEST (1.54-1.56) IS NOT EQUAL TO 999 AND CEMENTAL GLGS BEST (1.67-1.69) IS NOT TO 999 AND PULP CAVITY CONDITION (1.41) IS NOT EQUAL TO "3" ... CEMENTAL GLGS BEST MUST BE LESS THAN "1.6" TIMES DENTINAL GLGS BEST.
ERROR014	IF DENTINAL GLGS HIGH (1.57-1.59) IS NOT EQUAL TO 999 AND CEMENTAL GLGS HIGH (1.70-1.72) IS NOT EQUAL TO 999 AND PULP CAVITY CONDITION (1.41) IS NOT EQUAL TO "3" ... CEMENTAL GLGS HIGH MUST BE LESS THAN "1.6" TIMES DENTINAL GLGS HIGH.
ERROR015	IF DENTINAL GLGS LOW (1.60-1.62) IS NOT EQUAL TO 999 AND CEMENTAL GLGS LOW (1.73-1.75) IS NOT EQUAL TO 999 AND PULP CAVITY CONDITION (1.41) IS NOT EQUAL TO "3" ... CEMENTAL GLGS LOW MUST BE LESS THAN "1.6" TIMES DENTINAL GLGS LOW.
ERROR016	IF READER CODE (1.30-1.31) IS EQUAL TO "06" ... MICROSCOPE ID (1.39-1.40) MUST BE EQUAL TO "01".
ERROR017	IF READER CODE (1.30-1.31) IS EQUAL TO "18" ... MICROSCOPE ID (1.39-1.40) MUST BE EQUAL TO "02".
ERROR018	IF CONDITION OF SLIDE IS UNKNOWN (I.E. UNKNOWN (1.29) IS EQUAL TO "1") ... THE FOLLOWING ELEMENTS WHICH DESCRIBE CONDITION OF SLIDE MUST BE BLANK : FADED (1.23), SHRINKAGE (1.24), OFF CENTER (1.25), DARK (1.26), LIGHT (1.27), AND SHREDDED (1.28).
ERROR019	IF READER CODE (1.30-1.31) IS NON BLANK ... READER CODE MUST BE EQUAL TO EITHER "6" OR "18".
ERROR020	IF AGE RECORDED BEST (1.42-1.44) IS NOT EQUAL TO 999 ... AGE RECORDED BEST MUST BE LESS THAN 500.
ERROR021	IF AGE RECORDED HIGH (1.45-1.47) IS NOT EQUAL TO 999 ... AGE RECORDED HIGH MUST BE LESS THAN 500.
ERROR022	IF AGE RECORDED LOW (1.48-1.50) IS NOT EQUAL TO 999 ... AGE RECORDED LOW MUST BE LESS THAN 500.
ERROR023	IF DENTINAL GLGS BEST (1.54-1.56) IS NOT EQUAL TO 999 ... DENTINAL GLGS BEST MUST BE LESS THAN 500.
ERROR024	IF DENTINAL GLGS HIGH (1.57-1.59) IS NOT EQUAL TO 999 ... DENTINAL GLGS HIGH MUST BE LESS THAN 500.
ERROR025	IF DENTINAL GLGS LOW (1.60-1.62) IS NOT EQUAL TO 999 ... DENTINAL GLGS LOW MUST BE LESS THAN 500.
ERROR026	IF CEMENTAL GLGS BEST (1.67-1.69) IS NOT EQUAL TO 999 ... CEMENTAL GLGS BEST MUST BE LESS THAN 500.
ERROR027	IF CEMENTAL GLGS HIGH (1.70-1.72) IS NOT EQUAL TO 999 ... CEMENTAL GLGS HIGH MUST BE LESS THAN 500.
ERROR028	IF CEMENTAL GLGS LOW (1.73-1.75) IS NOT EQUAL TO 999 ... CEMENTAL GLGS LOW MUST BE LESS THAN 500.

RECOMMENDED ADDITION:

ERROR029 IF REQUEST REMOUNT/RECUT (1.32) IS ANSWERED YES (CODE 1 OR 2)...THIS READING IS NOT TO BE USED AS THE AGE ESTIMATE.

three females) remaining in the sample were removed because examination of the data on all spotted dolphins revealed that they were initially assigned the wrong stock code, i.e., they were mis-identified as northern offshore spotted dolphins by the observer (W. F. Perrin, pers. comm.).

Specimen "mix-ups", i.e. teeth from different animals assigned the same specimen number or those with incompatible age and life-history data, were searched for using a rigorous procedure. Initially, all of the readings (by both readers) available for a specimen were compared. For any series of counts that were notably different based on the age of the specimen and the quality of the preparations, the prepared thin sections for that specimen were examined together with the life-history data. Any aberrant thin sections were considered incorrect ("mix-ups"). The bad preparation, with the corresponding Tooth Reading Record and data-base record were each appropriately labeled. In cases where only two preparations were available, it was not always possible to distinguish which of the two was correct, so another slide was prepared for that specimen. This procedure requires that many more specimens be examined than are in error, but ensures that any errors in preparation are identified.

An additional specimen check was made after all the readings were completed and the tooth preparation mix-ups were rectified. Simple scatterplots of specimen length on age and number of corpora or testes weight on age showed outlying data points resulting from incompatible age and life history data. The outlying data points were identified by specimen number using the program OUTLIER (Appendix 5). Once the specimen number of outlying points was known, the life history record for the specimen was checked and a decision was made concerning the accuracy of the data.

TRSPTEDT Changes for Other Dolphin Stocks

Some changes in the range and logical error checks will be necessary to use the edit program on data for other dolphin stocks:

1. Preparation year (cols. 13-14).
2. Preparator code (cols. 19-20)
3. Reader code (cols. 30-31)
4. Reading year (cols. 33-34).
5. Stock code (cols. 77-78).
6. Reader code and matching microscope ID code (Logical Errors 17, 19 in Table 2).
7. Expected "age" ranges (Logical Errors 20-28).

In addition we recommend that the following logical error checks be added

CARD-NO	RECORD NUMBER=	3. *** 01.13-14 IS OUT OF RANGE.	
3	005RCD0044110809080111	06 80120501 999999999 2002101901 190999999102	*
8	011RCD0186118009080111	8. *** LOGICAL ERROR CHECK NUMBER ERROR002 FAILED	*
		06 80120501199999999 130999999 130999999 02	
CARD-NO	COLUMN	1-----2-----3-----4-----5-----6-----7-----8	
1	003WCF0134118009080111	06 8012050119999999999 9991301203 1309999999302	LOGICAL GROUP
2	004MTB0008118009120111	06 8012050119999999999 1609999992 160999999202	
3	005RCD0044110809080111	06 80120501 999999999 2002101901 190999999102	
4	006RCR0047118009120111	06 8012050119999999999 1309999993 130999999302	
5	007DRD0068118009120111	06 8012050119999999999 9991601504 150999999402	
6	008DBH0443118009080111	06 8012050119999999999 1509999994 150999999402	
7	009SWJ0075118009080111	06280120501 999999999 9999999996 190999999602	
8	011RCD0186118009080111	06 8012050119999999999 130999999 130999999 02	
9	012PAT0040118010060111	06 8012050119999999999 1509999994 150999999402	
10	013TCF0125118009120111	06280120501 999999999 999999999 260270260502	

TABLE 3. Output for one logical group from the Tooth Reading Record edit program (TRSPEDT).
 Two errors (boxed) were identified from this logical group, one in record number 3 and the other in record number 8. The type of error and the record containing the error are printed above the logical group. Record numbers (=card numbers) and column numbers are printed for each logical group.

to the edit program:

1. Include a logical check for cemental halves, similar to Errors 13-15 which currently check for cemental multiples.
2. Check for a valid reading of "age" (see Error 29, Table 2).

NAMING CONVENTION FOR THE INPUT AND OUTPUT FILES

A naming convention has been established for the input and output files created for and during the edit process. This is necessary because of the large number of files that are created and the different data sets and formats that are involved. The original data file, consisting of all the tooth reading records, is called "XXTRFSPT.ALL" or "XXTRMSPT.ALL", where

XX	First two characters, indicates that this is a non-indexed file
TR	for Tooth Reading
M or F	for male or female
SPT	for spotted dolphins (or, e.g., SPN for spinner dolphins for consistency with the other dolphin-data-base names)
.ALL	file type to indicate that all the records are included in this file.

When the file has been indexed (as are the other dolphin-data bases), the name of the file changes to TRFSPTDB.ALL. The XX is removed and DB (data base) is added to the end of the name. This name change is necessary since this version has been reformatted (see REFORMAT, Appendix 4) to place the primary (unique) key of Reader Code, Reading Date, and Reading of Day at the beginning of each record.

The file type in each data file name also changes during the edit process. As stated above, when the entire set of Tooth Reading Record data is in a single file, the file type is designated ".ALL". If some of the records must be removed because of specimen data errors ("mix-ups"), they are separated from the correct records by using the program GOODBAD on XXTRFSPT.ALL. The output file types from this program are ".AOK" for the correct records and ".BAD" for the bad records, i.e., XXTRFSPT.AOK and XXTRFSPT.BAD. Once the .AOK file is completed, it is reformatted, indexed, and renamed TRFSPTDB.DAT. This is the data file on which age-related analyses for female northern offshore spotted dolphins will be based.

DATA EXTRACTION PROGRAMS

Elements of the tooth-reading data were extracted as needed to create smaller "working" data files which may also combine elements from other data bases. The flowcharts for the spotted dolphin age data (Figures 2 and 3) illustrate all data-extraction programs and the resulting data files. In addition, DATATRIEVE domains and record descriptions have been created for many of the age-data files. Figure 4 gives a flowchart of these files.

The following is a list of data extraction programs (VAX Fortran) for use on the tooth-reading-data files and the resulting output data files. All data-file names mentioned here represent female spotted dolphins. For males, the "F" is replaced by "M".

	Program	Output Data File(s)
1.	GOOBBAD	XXTRFSPT.AOK XXTRFSPT.BAD
2.	LASTDATE	FSPTLAST.DAT
3.	MAGEBST	AGEBESTF.DAT
4.	REFORMAT	TRFSPTDB.DAT
5.	INDEX	(same as input file)
6.	MALE	(same as input file)
7.	MFSPLIT	XXFEM.DAT XXMALE.DAT
8.	MAKEDAT	FLHAGE.DAT
9.	MAKEDAT2	SLAGEyrF.DAT (where "yr" is last 2 digits of year)
10.	OUTLIER	(see description)

General descriptions of the spotted-dolphin-age-data extraction programs are provided below. Internal documentation is available in the programs which are listed in Appendices 4 and 5.

1. GOOBBAD program. After the original tooth-reading records are checked and edited, records from specimen "mix-ups" are marked with an asterisk ("*") in column 80. The GOOBBAD program separates the "bad" records, which will not be used in age analyses, from the "good" records. These "bad" data are stored in a file called XXTRFSPT.BAD. The remaining data are stored in the "good" file of age data, XXTRFSPT. AOK, on which all

FIGURE 2

SPOTTED DOLPHIN DATA FILES FLOWCHART: OVERVIEW

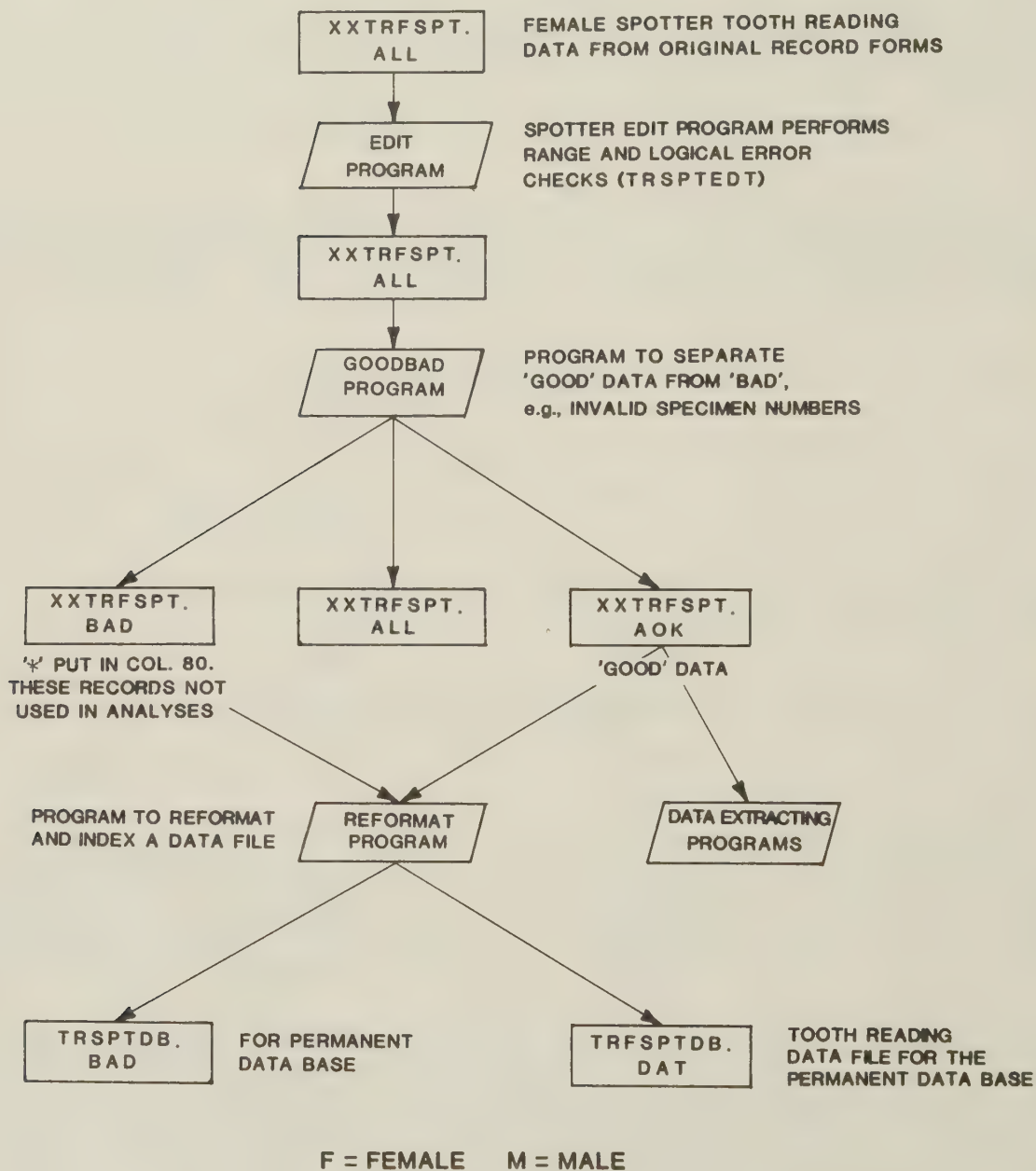


Figure 2. Spotted dolphin data files flowchart showing the steps used to generate the age data base. Although this chart illustrates the female data files, the procedure is the same for the male data, but the "F" would be replaced by an "M" in all data file names. The symbols are the same as in Figure 3.

SPOTTER AGE DATA FILES DETAILED FLOWCHART

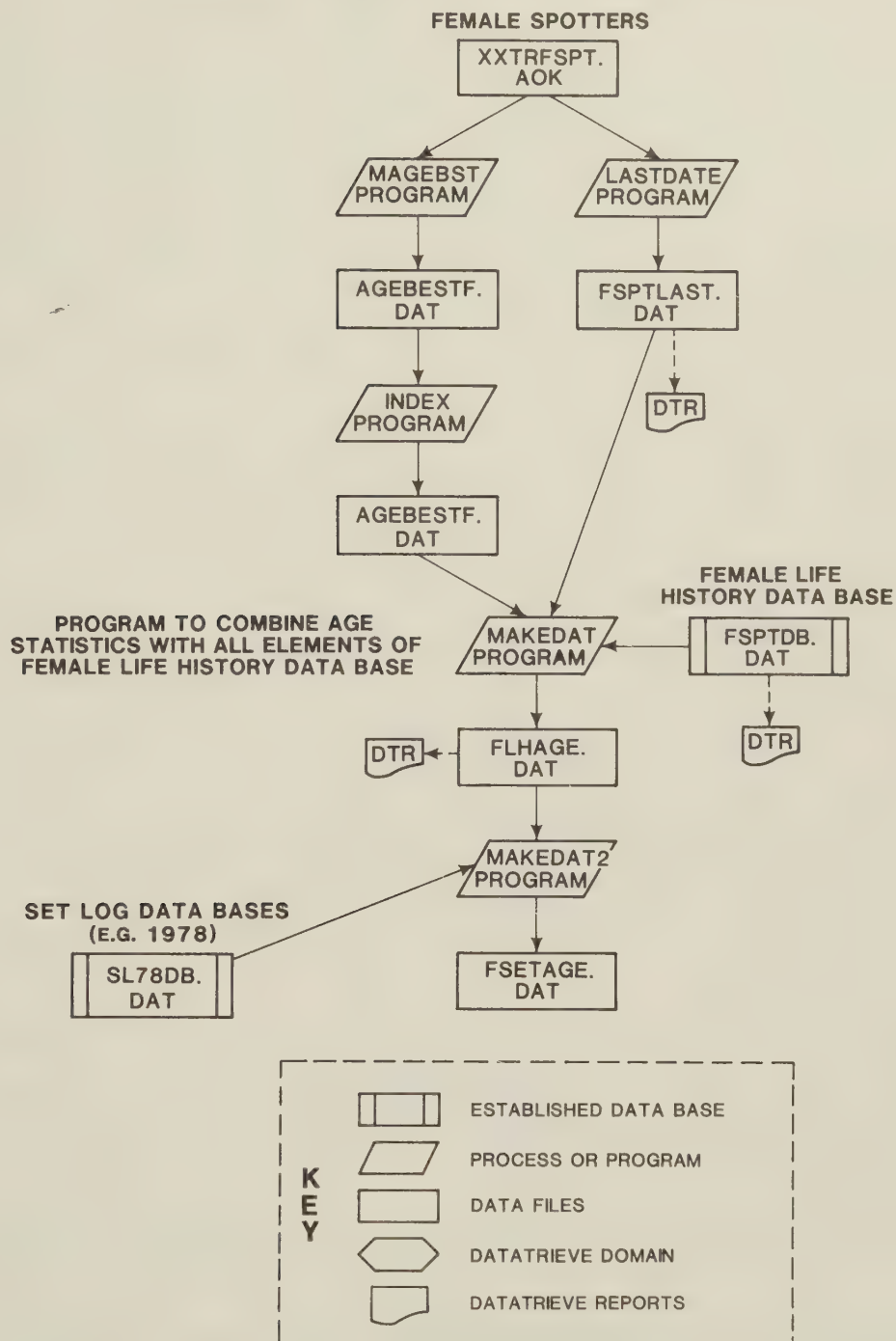


Figure 3. Spotted dolphin age data files detailed flowchart showing data extraction programs and input and output data files. Although this chart illustrates the female data files, the procedure is the same for the male files.

DATA RETRIEVE REPORTS

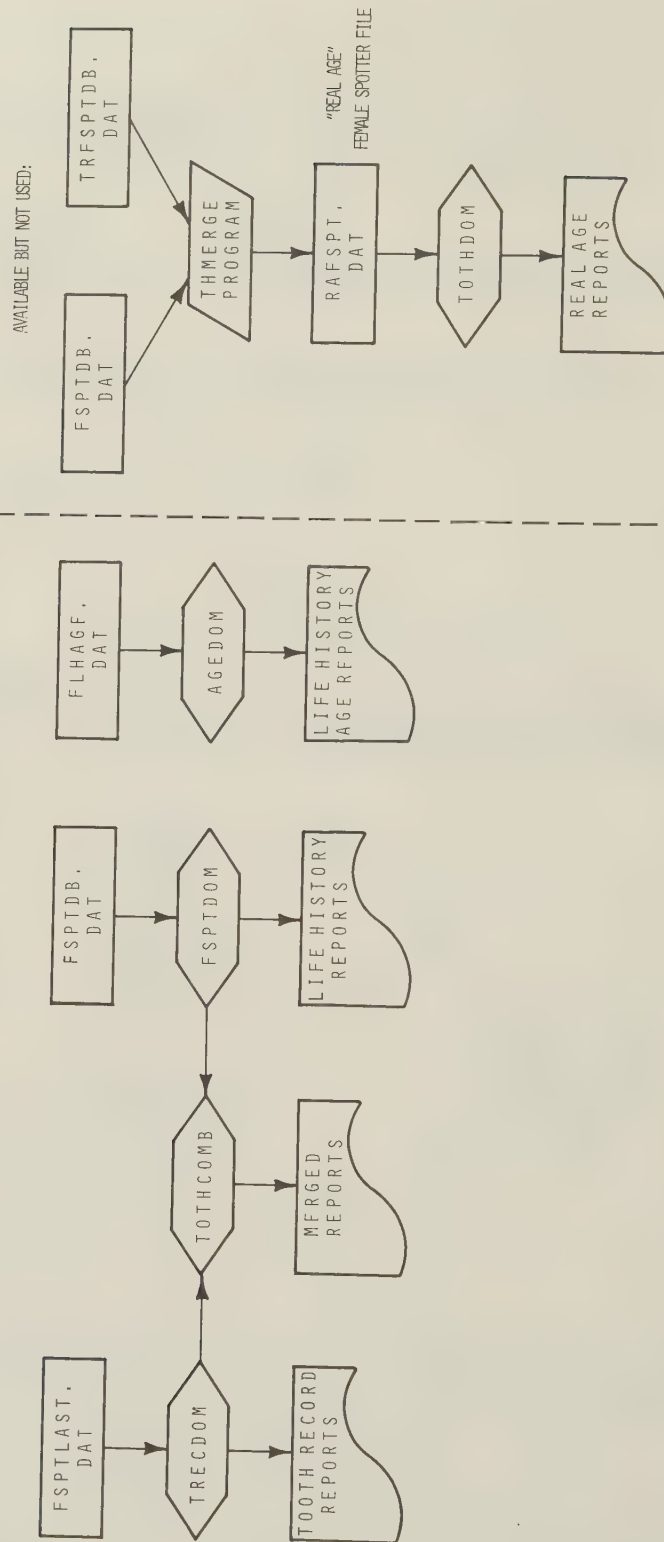


Figure 4. Flowchart showing data retrieve domains available for the age data and related data bases. Symbols are the same as in Figure 3.

age-related analyses are based.

2. LASTDATE program. For each specimen, the most recent (last) Tooth Reading Record in the data file for each reader is extracted to create FSPTLAST.DAT. Thus the "age" of each specimen as determined by the most recent age reading is available in a separate file. This file can be indexed and reformatted to generate DATATRIEVE reports.
3. MAGEBST program. This program calculates means and variances using "Age Recorded Best" counts in XXTRFSPT.AOK for a precision analysis of age counts (Reilly et al., 1983). The resulting data file, AGEBESTF.DAT, contains statistics such as mean "age" per specimen by reader, a pooled mean age for each specimen, and variance estimates.
4. REFORMAT program. This program reformats and indexes a tooth-reading data file for consistency and compatability with other established data bases. The "User's Guide for the Tooth Reading Edit Process" in Appendix 3 fully describes the logic and use of the REFORMAT program.
5. INDEX program. The INDEX program indexes an age-data file without any reformatting of the records. The user may edit the program to change the input file, record size, or location of the indexing key. The INDEX program was used to index AGEBESTF.DAT before running the MAKEDAT program which requires all input data files to be indexed for matching records by specimen number between the input data files.
6. MALE and MFSPLIT programs. If an analysis requires the merging of female and male spotter age data, the MALE program is available to mark all male data with a "1" in column 79. Later the records for each sex can be separated using the MFSPLIT program, which stores female records in XXFEM.DAT and male records in XXMALE.DAT. (The "1" will remain in column 79).
7. MAKEDAT program. Age data (i.e., reader mean ages, pooled mean age, last age recorded "best", "high", and "low" for each reader) are appended to all elements of the female life-history data base for each specimen in the age-determination subsample to create a large indexed file, FLHAGE.DAT. DATATRIEVE reports can be generated with this file.

Non-matching specimen numbers between each of the input data files are printed out during execution to allow for error correction in both of the MAKEDAT programs.

The MAKEDAT programs may be used with the spinner dolphin (or other) life-history data base by changing the input file from FSPTDB.DAT to FSPNDB.DAT.

8. MAKEDAT2 program. Age, life-history, and set-log data are combined to create a SETAGEyr.DAT file for each year. The resulting file provides NMFS observer data such as dolphin school size, set conditions, geographic area, and known dolphin kill.

9. OUTLIER program. The OUTLIER program identifies outlying data points, i.e., those which lie outside the given or expected value defined in the program. The user can easily edit an outlier program to specify any range of values to examine. The "WRITE" statement can also be altered to output additional variables for each specimen. Outlier programs are most useful for isolating suspected "bad" data points. This version of the OUTLIER program prints the specimen number of female spotted dolphins within a given range of "age" and corpora count based on data from FLHAGE.DAT (see Appendix B). In this example, a scatterplot of corpora on age produced a number of outlying data points. These points (specimens) can be checked by specifying their coordinates in the OUTLIER program "IF" statements. The first "IF" statement looks for specimens within a given range of age and corpora count. (ACMLAVG, AAHLAVG and POOLM are three separate determinations of "age".) The user has programmed for records with pooled-mean-age less than 10 with a corpora count greater than zero. These specimen numbers will be printed out along with an "IVAL" equal to 100. The specimen can then be checked in the life-history data records.

ACKNOWLEDGEMENTS

The Tooth Reading Record was developed by the first author, Frank Ralston, Bruce Wahlen, and Al Myrick, OFRD, and prepared by Roy Allen, SWFC Illustrator. Roy Allen also drafted the Age Data flowcharts. Most of the FORTRAN programs were written by Jay Walker, SWFC. Susan Boyer, contractor, wrote MAGEBST. Frank Ralston was instrumental in making the Tooth Reading edit programming procedure, including use of logical error checks, similar to the other dolphin data bases. Susan Chivers helped with data extraction and in the compilation of this report. Al Myrick, Frank Ralston, Michael Scott, Ken Wallace, and Bruce Wahlen provided useful comments on the manuscript.

LITERATURE CITED

- Myrick, A. C., Jr., A. A. Hohn, P. A. Sloan, M. Kimura, and D. Stanley. 1983. Estimating age of spotted and spinner dolphins (Stenella attenuata and Stenella longirostris) from teeth. NOAA-TM-NMFS-SWFC-30., 17 pp.
- Perrin, W. F., J. M. Coe, and J. R. Zweifel. 1976. Growth and reproduction of the spotted porpoise, Stenella attenuata, in the offshore eastern tropical Pacific. Fish. Bull. (U.S.) 74(2):229-269.
- Perrin, W. F. and A. C. Myrick, Jr. (eds.) 1980[1981]. Age determination of toothed whales and sirenians. Rep. int. Whal. Commn. (Special Issue No. 3), 229 pp.

APPENDIX 1

Tooth Reading Record Usage Notes

TOOTH READING RECORD USAGE NOTES

Introduction

In addition to recording dentinal and cemental GLG counts and specimen ages, the Tooth Reading Record has been designed to characterize the preparation, reading, and quality of sections used for age determination.

1. For clarity, the Tooth Reading format has been divided into 8 sections:

- Specimen, General
- Condition of Slide
- Reading, General
- Age Recorded
- Dentinal
- Cemental
- Code Tables
- Notes

2. Data for "Specimen, General", except code blocks 1.1-3, are noted by preparator on the slide. However, data for all sections are to be encoded by the reader during the reading of each slide.
3. Every slide examined should have a Tooth Reading Record, regardless of whether or not an age was determined. If a slide is unacceptable, code block 1.33, "Request Remount/Recut", will be answered "Yes" (i.e., 1 or 2) and the slide will be returned to the preparator. However, each Record will be completed as fully as possible and included in the data set of Tooth Reading Records.
4. The first 3 code blocks of Specimen No., i.e., 1.4-6, and the notes section are the only sections where the coder may enter alphabetic characters. The remainder of the code blocks consist of numeric characters only.
5. A notes section has been added to the Tooth Reading Record to allow for hand-printed additional notes to be made by the reader. These notes, although not keypunched, may be useful in future evaluation of teeth and age counts.
6. An asterisk on the Coding Format and Tooth Reading Record indicates elements whose fields must not be left blank. These are:

- Reading of Day
- Specimen No.
- Reader Code
- Reading Date
- Age Recorded: Best, High and Low
- Number of Dentinal GLGs: Best, High and Low
- Number of Cemental GLGs: Best, High and Low

7. All numeric values coded should be right-justified within a field. Leading zeros are not required unless specified.
8. No/unrecorded responses:
 - a. For a Yes/No question - "No" may be coded as a single blank or "0".
 - b. "Unrecorded"
 - i) For the nine elements for which "Ø" is a valid value (i.e., Age Recorded - Best, High, Low; Number of Dentinal GLGs - Best, High, Low; Number of Cemental GLGs - Best, High, Low), the filler 999 should be coded when data is not available or unrecorded.
 - ii) For all remaining elements, except those which must receive a value (see 6), "unrecorded" data is indicated by leaving the field blank.

TOOTH READING RECORD
CODING FORMAT

Columns	Title	Explanation or Reference
1-3	*Reading of day	Up to 3-digit number indicating the serial position of the specimen within a day's sequence.
4-10	*Specimen No.	Observer's initials (up to 3) are left-justified alphabetic characters. The remaining 4 characters are right-justified numeric characters with leading zeros.
11	Tooth of specimen	For a slide, the number of times different teeth from an animal have been prepared, i.e., 1= original tooth, 2= sections are from the second tooth prepared from that animal.
12	Slide of tooth	The mount number, i.e., the mount being read in a series of mounts for the same tooth.
13-14	Preparation date: Year	The last 2 digits of the year in which the slide was added to the total collection of slides to be read.
15-16	Preparation date: Month	The month in which the slide was added to the total collection of slides to be read. 2 digits.
17-18	Preparation date: Day	The day in which the slide was added to the total collection of slides to be read. 2 digits.
19-20	Preparator	The coded ID of the person who prepared the slide. (Code Table 1).
21	Tooth treatment	The coded method of preparation of the tooth. (Code Table 2).
22	Mounting medium	The coded material used to mount the sections on the slide. (Code Table 3).
23	Faded	The coded answer to the question, "Were the sections on a slide faded?" Yes = 1, No/unrecorded = 0 or blank. (Code Table 4 for columns 23-29).

24	Shrinkage	The coded answer to the question, "Were the sections on the slide shrunken?" Yes = 1, No/unrecorded = 0 or blank.
25	Off-center	The coded answer to the question, "Were the sections on the slide cut off-center?" Yes = 1, No/unrecorded = 0 or blank.
26	Dark	The coded answer to the question, "Were the sections on the slide too dark?" Yes = 1, no/unrecorded = 0 or blank.
27	Light	The coded answer to the question, "Were the sections on the slide too light?" Yes = 1, No/unrecorded = 0 or blank.
28	Shredded	The coded answer to the question, "Were the sections on the slide shredded?" Yes = 1, No/unrecorded = 0 or blank.
29	Unknown	The coded answer to the question, "Were the slides of poor quality for non-defined or not recorded reasons?" Yes = 1, No/unrecorded = 0 or blank.
30-31	*Reader Code	The unique 2-digit number assigned to the reader of the slide. (Code Table 5).
32	Request remount/ recut	The coded answer to the questions, "Was a remount or recut requested by the reader for this slide? If so, which one?" (Code Table 7).
33-34	*Reading date: Year	The year in which this slide was read. 2 digits.
35-36	*Reading date: Month	The month in which slide was read. 2 digits.
37-38	*Reading date: Day	The day on which this slide was read. 2 digits.
39-40	Microscope ID	The 2-digit code indicating type of microscope used to read this slide. (Code Table 8).
41	Pulp cavity condition	The 1-digit code indicating the degree of closure of the pulp cavity. (Code Table 9).
42-44	*Age recorded: Best	The 3-digit number rounded to the nearest tenth which records the best estimate of the age of the animal. Unrecorded = 999.

45-47	*Age recorded: High	The 3-digit number rounded to the nearest tenth which records the highest estimate in a range of estimates for the age of the animal. Unrecorded = 999.
48-50	*Age recorded: Low	The 3-digit number rounded to the nearest tenth which records the lowest estimate in a range of estimates for the age of the animal. Unrecorded = 999.
51	Dentinal GLG condition: Indistinct	The coded answer to the question, "Were the GLGs indistinct, meaning that the GLGs are not or scarcely distinguishable in all or part of a section?" Yes = 1, No/unrecorded = 0 or blank.
52	Dentinal GLG condition: Irregular	The coded answer to the question, "Was the dentine irregular, meaning that GLGs change from regular cone shaped into a wavy or otherwise non-uniform pattern?" Yes = 1, no/unrecorded = 0 or blank.
53	Dentinal GLG condition: Pearls	The coded answer to the question, "Were there pearls, meaning areas of globular dentine disrupting GLGs, in the dentine?" Yes = 1, no/unrecorded = 0 or blank.
54-56	*Number of dentinal GLGs: Best	The 3-digit number rounded to the nearest tenth which records the best estimate of the number of dentinal GLGs. Unknown/not recorded = 999.
57-59	*Number of dentinal GLGs: High	The 3-digit number rounded to the nearest tenth which records the highest estimate in a range for the number of dentinal GLGs. Unknown/unrecorded = 999.
60-62	*Number of dentinal GLGs: Low	The 3-digit number rounded to the nearest tenth which records the lowest estimate in a range for the number of dentinal GLGs. Unknown/unrecorded = 999.
63	Confidence of Dentinal reading	The coded answer indicating the determination by the reader on the level of confidence reader had in the dentinal GLG count made. (Code Table 6).
64	Cemental GLG condition: Indistinct	The coded answer to the question, "Were the cemental GLGs indistinct, meaning that the GLGs are not or scarcely distinguishable in all or part of a section?" Yes = 1, No/unrecorded = 0 or blank.

65	Cemental GLG condition: Multiples	The coded answer to the question, "Did the cemental GLGs occur as multiples (2x, 3x...) of the number of dentinal GLGs counted?" Yes = 1, No/unrecorded = 0 or blank.
66	Cemental GLG condition: Halves	The coded answer to the question, "Did the cemental GLGs appear as half of the number of dentinal GLGs counted?" Yes = 1, No/unrecorded = 0 or blank.
67-69	*Number of cemental GLGs: Best	The 3-digit number to the nearest tenth which records the best estimate of the number of cemental GLGs which represents the age. Unknown/unrecorded = 999.
70-72	*Number of cemental GLGs: High	The 3-digit number to the nearest tenth which records the highest estimate in a range for the number of cemental GLGs which represents the age. Unknown/unrecorded = 999.
73-75	*Number of cemental GLGs: Low	The 3-digit number to the nearest tenth which records the lowest estimate in a range for the number of cemental GLGs which represents the age. Unknown/unrecorded = 999.
76	Confidence of Cemental count	The coded answer indicating the determination by the reader on the level of confidence reader had in the cemental GLG count made. (Code Table 6).
77-78	Dolphin stock	The coded ID of the species and stock of dolphins which the teeth were taken.

*Must not be left blank.

TOOTH READING RECORD

CODE TABLES

Code Table With DefinitionsExplanation

CODE TABLE 1: PREPARATOR (1.19-20)

01	Sloan	04	Hohn
02	Kimura	05	Bioanalysis Inc.
03	Stanley	06	Perrin et al

Assigns a 2-digit code to identify the person who prepared a slide.

CODE TABLE 2: TOOTH TREATMENT (1.21)

1	Stained	- whole tooth decalcified, sections cut then stained in hematoxylin before mounting
2	Unstained	- undecalcified tooth section

Assigns a 1-digit code for the method of preparation of the sections on a slide.

CODE TABLE 3: MOUNTING MEDIUM (1.22)

1	Permount
2	Glycerin jelly
3	Glycerin

Assigns a 1-digit code to identify the material used to mount the section on a slide.

CODE TABLE 4: CONDITION OF SLIDE (1.23-29)

Ø or blank = No/unrecorded

1 = Yes

1.23	Faded -	fading of stain from a decalcified and stained section. May or may not be uniform for all the sections on a slide or within a section. Results in lack of clarity of GLGs.
1.24	Shrinkage -	section smaller than its counterparts which have not been mounted or decalcified; decrease in size due to shrinkage of the tissue.
1.25	Off-center-	mounted sections not a mid-longitudinal cut from the tip of the tooth through the center of the pulp cavity and root.
1.26	Dark -	for stained sections, sections over-stained. For unstained sections, too dense, inadequate light transmitted.
1.27	Light -	for stained sections, sections under-stained or possibly due to leaching out of stain evenly across a section. For unstained sections, too thin, too much light transmitted.
1.28	Shredded -	tearing or ripping of the tissue caused during the cutting process. May also refer to when the tissue is torn only on the surface, not all the way through, over a wide area, destroying the GLGs.
1.29	Unknown -	The sections are poor or unacceptable for reasons not specified.

Assigns a coded answer to a series of characteristics that can be found when examining a slide of tooth sections. Yes means that the condition occurs. Blank or no indicates that the condition was not recorded. If all 7 blocks are blank, the slide is good. If Yes for Unknown, columns 1.23-28 should be left blank.

CODE TABLE 5: READER CODE (1.30-31)

01	Perrin	11	Seagars
02	Holts	12	Gurevich
03	Clapp	13	Mead
04	Suanico	14	Ode11
05	Coe	15	Stuart
06	Myrick	16	Kasuya
07	Henderson	17	Brownell
08	Kimura	18	Hohn
09	Sloan	19	
10	Hui	20	

Assigns a 2-digit code to identify the person who reads the slide.

APPENDIX 2

**VAX Fortran Northern
Offshore Spotted
Dolphin Edit Program**

```

C*****
C*****
C
C* PROGRAM:TRSPTEDT
C
C* PURPOSE: TO EDIT THE TOOTH READING DATAFILE
C
C* PROGRAMMER: G. JAY WALKER
C
C* LANGUAGE: FORTRAN(VAX)
C
C* DATE:FEBRUARY 1982
C
C*****
C*****
C
C* BASIC PROCESSING LOGIC - THE PROGRAM WILL CALL THE SUBROUTINE CSOOTR WHICH
C* CHECKS TO SEE THAT THE CARDS ARE IN PROPER SEQUENCE (I.E., SORTED BY
C* READER CODE, READER DATE, READING OF DAY). IF NO SEQUENCING ERRORS
C* ARE FOUND, THE EDIT PROGRAM CONTINUES. A RECORD IS READ IN AND
C* GOES THROUGH A SERIES OF SINGLE FIELD CHECKS (E.G., WHETHER A FIELD
C* IS BLANK, WHETHER IT IS WITHIN CERTAIN RANGE VALUES). THEN A SERIES
C* OF INTERFIELD CHECKS IS PERFORMED. ANY ERRORS ARE PRINTED OUT.
C* WHEN THE END OF A LOGICAL RECORD GROUP IS REACHED, THE RECORDS
C* IN THE GROUP ARE PRINTED OUT. (A LOGICAL GROUP CONTAINS A SET OF
C* RECORDS WITH THE SAME READING CODE AND READING DATE, UP TO 10 RECORDS.)
C
C* INPUT/OUTPUT - THE USER IS PROMPTED FOR THE NAME OF THE INPUT FILE
C* WHICH WILL BE THE PARTICULAR TOOTH READING DATA TO BE EDITED.
C* OUTPUT IS A LISTING OF THE ERRORS FOUND.
C
C* COMPILING AND LINKING - TO COMPILE THE PROGRAM USE:
C*      $FOR TRSPTEDT
C* TO LINK THE PROGRAM USE:
C*      $LINK TRSPTEDT,CSOOTR,BLANKT,CSEQ,EJECT,ERRCHK,ERRLOG,GENER1,
C*      GENER2,GENER3,P,PGROUP,RANGIT,VALUIT,VERFIT,WRAPUP
C
C* EXECUTION - TO EXECUTE THE PROGRAM USE:
C*      $SUBMIT TRSPTEDT
C* TO EDIT A DIFFERENT TOOTH READING DATAFILE: USE THE VAX EDITOR TO
C* EDIT THE COMMAND FILE TRSPTEDT.COM . CHANGE THE FOURTH LINE OF THIS
C* COMMAND FILE TO THE NAME OF THE TOOTH READING DATAFILE TO BE EDITED
C* (NOTE - THE COMMAND FILE IS SET FOR GO EDIT (AS OPPOSED TO NO EDIT)
C* AND RESPONDS TRUE TO THE QUESTION OF WHETHER TO PRINT ALL DATA GROUPS)
C
C* SUBROUTINES - THE PROGRAM USES THE SET OF COMMON SUBROUTINES. SOME
C* OF THESE SUBROUTINES ARE AS FOLLOWS:
C*   BLANKT - VERIFIES THAT A FIELD IS NON-BLANK
C*   EJECTB,EJECTH,EJECTR - DEALS WITH PAGE HEADINGS AND LINE NUMBERING
C*   ERRLOG - OBTAINS AN ERROR MESSAGE FROM THE ERROR DATABASE
C*   CLLGCT - CLOSSES THE ERROR DATABASE
C*   OPLGCT - OPENS THE ERROR DATABASE
C*   P(ARG) - FUNCTION SUBPROGRAM WHICH CHECKS THE ARGUMENT FOR NUMERIC DATA
C*   AND ZERO FILLS IT TO THE LEFT IF IT IS VALID NUMERIC
C*   RANGIT - VERIFIES THE VALUE OF A FIELD FALLS WITHIN A SPECIFIED RANGE

```

```

C*   VERFIT - VERIFIES THAT ALL CHARACTERS OF A STRING ARE CHARACTERS
C*   OF ANOTHER STRING
C*   WRAPUP - PRINTS OUT A LOGICAL RECORD GROUP
C
C*   VARIABLES - FOLLOWING IS A PARTIAL LIST OF VARIABLES USED BY THE PROGRAM:
C*   ALLBLK - USER-ENTERED LOGICAL VARIABLE, SET TO TRUE IF ALL DATA
C*   BLOCKS ARE TO BE PRINTED
C*   ANYERR - SET TO TRUE ON THE OCCURANCE OF ANY ERROR
C*   CURRSTR - THE CURRENT RECORD
C*   DSKSEQ - CONTAINS THE CURRENT PHYSICAL RECORD NUMBER
C*   ELEVAl - CODE USED TO SPECIFY THE AMOUNT OF DETAIL IN THE ERROR MESSAGE
C*   FLAG - WILL HAVE A VALUE OF 1 AFTER THE FIRST RECORD IS FINISHED
C*   BEING PROCESSED
C*   HEAD - CONTAINS THE PAGE HEADING CHARACTER STRING
C*   INFILE - CONTAINS THE NAME OF THE INPUT FILE
C*   KOUNT - COUNT OF THE NUMBER OF RECORDS IN THE CURRENT LOGICAL GROUP
C*   LASSEQ - NUMBER OF PHYSICAL RECORDS ON THE DISK FILE
C*   LOGGRP - LOGICAL GROUP NUMBER IDENTIFIER
C*   STORE - AN ARRAY WHICH CONTAINS THE RECORDS IN THE CURRENT LOGICAL
C*   RECORD GROUP
C
C*****
C
COMMON/CERROR/ CARSEQ,DATSET,GRPCOD,STRING,TEMP,INFILE
COMMON/ERROR/ ANYERR,DSKSEQ,ELEVAl,NOGO,VALUE,TOTERR,ALLBLK
CHARACTER GRPCOD*1,STRING*80,TEMP*80,DATSET*8,CARSEQ*2,INFILE*20,
1      PREVSTR*80,CURRSTR*80,STORE(10)*80,HEAD*60,P*8,FSTRX*6,
2      NUMST*6,GOEDIT*7,VAR1*8,VAR2*8,CHAR1*3,CHAR2*3,
3      ALPHAS*26/'ABCDEFGHIJKLMNOPQRSTUVWXYZ'/,
4      DIGITS*10/'0123456789'/
INTEGER DSKSEQ,ELEVAl,VALUE,FBRKCX,STARTC
DATA ELEVAl/01/, GRPCOD/' '/
C
C* SECTION 1
C* BEGIN MAIN PROGRAM LOGIC
C
CALL CSOOTR
ELEVAl=1
CALL OPLGCT('ECFSTR. ')
CALL EJECTR(1)
C
C* PROMPT USER FOR GO/NOGO EDIT
C
WRITE(6,3)
3 FORMAT(' ENTER 'GO EDIT' OR 'NO EDIT'.')
READ(5,4)GOEDIT
4 FORMAT(A7)
IF (GOEDIT .EQ. 'GO EDIT') GO TO 7
IF (GOEDIT .EQ. 'NO EDIT') GO TO 999
WRITE(6,5)GOEDIT
5 FORMAT(' RESPONSE - ',A7,' NOT 'GO EDIT' OR 'NO EDIT'.')
GO TO 999
7 CONTINUE
C
C* OPEN THE DATABASE

```



```

C      OPEN (UNIT=10,FILE=INFILE,TYPE='OLD')
      DATSET(1:8)= 'TRSP TEDT'
C
C* PROMPT USER FOR VALUE OF ALLBLK
C      WRITE(6,8)
      8 FORMAT(' ENTER TRUE/FALSE PRINT ALL DATA GROUPS. ')
      READ(5,*)ALLBLK
      LASSEQ=DSKSEQ
      DSKSEQ=0

C
C* HEAD UP FIRST OUTPUT PAGE
C      STARTC=1
      HEAD= '*** TRSP TEDT---INPUT FILE: '//INFILE(1:12)//
      1 'LOGICAL GROUP '//NUMST(STARTC: )

      CALL EJECTH(HEAD)
      CALL EJECTB(54)

C
C* READ FIRST CARD
C      READ(10,9)CURRSTR
      9 FORMAT(A80)
      LOGGRP=1
      KOUNT=1
      STORE(KOUNT)=CURRSTR

C
C* SECTION 2
C* BLANKNESS AND RANGE CHECKS
C      50 CARSEQ= '01'
      DSKSEQ= DSKSEQ+1
      STRING=CURRSTR

C      CALL BLANKT(1,3)
      CALL RANGIT(1,3,1,150)

C      CALL BLANKT(4,3)
      IF (NOGO) GO TO 53
      CALL VERFIT(4,3,ALPHAS)

C      53 CALL BLANKT(7,4)
      IF (NOGO) GO TO 54
      CALL VERFIT(7,4,DIGITS)
      54 CALL RANGIT(7,4,1,999)

C      CALL RANGIT(11,1,1,9)

C      CALL RANGIT(12,1,1,9)

C      CALL RANGIT(13,2,79,83)

C      CALL RANGIT(15,2,0,12)
C

```

```

CALL RANGIT(17,2,0,31)
C
CALL RANGIT(19,2,1,6)
C
CALL RANGIT(21,1,1,2)
C
CALL RANGIT(22,1,1,3)
C
CALL RANGIT(23,1,0,1)
C
CALL RANGIT(24,1,0,1)
C
CALL RANGIT(25,1,0,1)
C
CALL RANGIT(26,1,0,1)
C
CALL RANGIT(27,1,0,1)
C
CALL RANGIT(28,1,0,1)
C
CALL RANGIT(29,1,0,1)
C
CALL BLANKT(30,2)
CALL RANGIT(30,2,6,18)
C
CALL RANGIT(32,1,1,3)
C
CALL BLANKT(33,2)
IF (NOGO) GO TO 55
CALL VERFIT(33,2,DIGITS)
55 CALL RANGIT(33,2,79,83)
C
CALL BLANKT(35,2)
IF (NOGO) GO TO 56
CALL VERFIT(35,2,DIGITS)
56 CALL RANGIT(35,2,1,12)
C
CALL BLANKT(37,2)
IF (NOGO) GO TO 57
CALL VERFIT(37,2,DIGITS)
57 CALL RANGIT(37,2,1,31)
C
CALL RANGIT(39,2,1,2)
C
CALL RANGIT(41,1,1,3)
C
CALL BLANKT(42,3)
CALL RANGIT(42,3,0,999)
C
CALL BLANKT(45,3)
CALL RANGIT(45,3,0,999)
C
CALL BLANKT(48,3)
CALL RANGIT(48,3,0,999)
C

```

```

      CALL RANGIT(51,1,0,1)
C
      CALL RANGIT(52,1,0,1)
C
      CALL RANGIT(53,1,0,1)
C
      CALL BLANKT(54,3)
      CALL RANGIT(54,3,0,999)
C
      CALL BLANKT(57,3)
      CALL RANGIT(57,3,0,999)
C
      CALL BLANKT(60,3)
      CALL RANGIT(60,3,0,999)
C
      CALL RANGIT(63,1,1,6)
C
      CALL RANGIT(64,1,0,1)
C
      CALL RANGIT(65,1,0,1)
C
      CALL RANGIT(66,1,0,1)
C
      CALL BLANKT(67,3)
      CALL RANGIT(67,3,0,999)
C
      CALL BLANKT(70,3)
      CALL RANGIT(70,3,0,999)
C
      CALL BLANKT(73,3)
      CALL RANGIT(73,3,0,999)
C
      CALL RANGIT(76,1,1,6)
C
      CALL BLANKT(77,2)
      CALL RANGIT(77,2,2,2)
C
C* SECTION 3
C* BEGIN INTER-VARIABLE LOGIC CHECKS
C
C* SKIP FIRST 2 ERRORS IF FIRST CARD
C
      IF (FLAG.EQ.1) GO TO 105
      FLAG=1
      GO TO 125
C
C* ERROR001
C
      105 IF (CURRSTR(30:31) .EQ. ' ' ) GO TO 110
      IF (CURRSTR(30:31) .EQ. PREVSTR(30:31)) GO TO 110
      CALL ERRLOG ('ERROR001')
      110 CONTINUE
C
C* ERROR002
C

```



```

      IF (CURRSTR(33:38) .NE. PREVSTR(33:38)) GO TO 120
      CHAR1=CURRSTR(1:3)
      CHAR2=PREVSTR(1:3)
      READ(CHAR1,'(I3)')INT1
      READ(CHAR2,'(I3)')INT2
      IF (INT1 .NE. (INT2+1)) THEN
        CALL ERRLOG ('ERROR002')
      END IF

```

```

120 CONTINUE

```

```

C
C* ERROR003

```

```

C
125 IF (CURRSTR(13:18) .EQ. '      ') GO TO 130
    IF (CURRSTR(33:38) .EQ. '      ') GO TO 130
    IF (CURRSTR(13:18) .LT. CURRSTR(33:38)) GO TO 130
    CALL ERRLOG ('ERROR003')
130 CONTINUE

```

```

C
C* ERROR004

```

```

C
    IF ((CURRSTR(42:44) .EQ. '999') .OR. (CURRSTR(45:47) .EQ. '999'
1)) GO TO 140
    IF (CURRSTR(45:47) .GE. CURRSTR(42:44)) GO TO 140
    CALL ERRLOG ('ERROR004')
140 CONTINUE

```

```

C
C* ERROR005

```

```

C
    IF ((CURRSTR(42:44) .EQ. '999') .OR. (CURRSTR(48:50) .EQ. '999'
1)) GO TO 150
    IF (CURRSTR(48:50) .LE. CURRSTR(42:44)) GO TO 150
    CALL ERRLOG ('ERROR005')
150 CONTINUE

```

```

C
C* ERROR006

```

```

C
    IF ((CURRSTR(45:47) .EQ. '999') .OR. (CURRSTR(48:50) .EQ. '999'
1)) GO TO 160
    IF (CURRSTR(45:47) .GT. CURRSTR(48:50)) GO TO 160
    CALL ERRLOG ('ERROR006')
160 CONTINUE

```

```

C
C* ERROR007

```

```

C
    IF ((CURRSTR(54:56) .EQ. '999') .OR. (CURRSTR(57:59) .EQ. '999'
1)) GO TO 170
    IF (CURRSTR(57:59) .GE. CURRSTR(54:56)) GO TO 170
    CALL ERRLOG ('ERROR007')
170 CONTINUE

```

```

C
C* ERROR008

```

```

C
    IF ((CURRSTR(54:56) .EQ. '999') .OR. (CURRSTR(60:62) .EQ. '999'
1)) GO TO 180
    IF (CURRSTR(60:62) .LE. CURRSTR(54:56)) GO TO 180

```

```

        CALL ERRLOG ('ERROR008')
180 CONTINUE
C
C* ERROR009
C
        IF ((CURRSTR(57:59) .EQ. '999') .OR. (CURRSTR(60:62) .EQ.
1'999')) GO TO 190
        IF (CURRSTR(57:59) .GT. CURRSTR(60:62)) GO TO 190
        CALL ERRLOG ('ERROR009')
190 CONTINUE
C
C* ERROR010-
C
        IF ((CURRSTR(67:69) .EQ. '999') .OR. (CURRSTR(70:72) .EQ.
1'999')) GO TO 200
        IF (CURRSTR(70:72) .GE. CURRSTR(67:69)) GO TO 200
        CALL ERRLOG ('ERROR010')
200 CONTINUE
C
C* ERROR011
C
        IF ((CURRSTR(67:69) .EQ. '999') .OR. (CURRSTR(73:75) .EQ.
1'999')) GO TO 210
        IF (CURRSTR(73:75) .LE. CURRSTR(67:69)) GO TO 210
        CALL ERRLOG ('ERROR011')
210 CONTINUE
C
C* ERROR012
C
        IF ((CURRSTR(70:72) .EQ. '999') .OR. (CURRSTR(73:75) .EQ.
1'999')) GO TO 220
        IF (CURRSTR(70:72) .GT. CURRSTR(73:75)) GO TO 220
        CALL ERRLOG ('ERROR012')
220 CONTINUE
C
C* ERROR013
C
        IF ((CURRSTR(54:56) .EQ. '999') .OR. (CURRSTR(67:69) .EQ.
1'999') .OR. (CURRSTR(41:41) .EQ. '3')) GO TO 230
        CHAR1=CURRSTR(54:56)
        CHAR2=CURRSTR(67:69)
        READ(CHAR1, '(I3)')INT1
        READ(CHAR2, '(I3)')INT2
        IF (INT2 .LT. 1.6*INT1) GO TO 230
        CALL ERRLOG ('ERROR013')
230 CONTINUE
C
C* ERROR014
C
        IF ((CURRSTR(57:59) .EQ. '999') .OR. (CURRSTR(70:72) .EQ.
1'999') .OR. (CURRSTR(41:41) .EQ. '3')) GO TO 240
        CHAR1=CURRSTR(57:59)
        CHAR2=CURRSTR(70:72)
        READ(CHAR1, '(I3)')INT1
        READ(CHAR2, '(I3)')INT2

```

```

        IF (INT2 .LT. 1.6*INT1) GO TO 240
        CALL ERRLOG ('E0OR0014')
240 CONTINUE
C
C* ERROR015
C
        IF ((CURRSTR(60:62) .EQ. '999') .OR. (CURRSTR(73:75) .EQ. '999')
1.OR. (CURRSTR(41:41) .EQ. '3')) GO TO 250
        CHAR1=CURRSTR(60:62)
        CHAR2=CURRSTR(73:75)
        READ(CHAR1,'(I3)')INT1
        READ(CHAR2,'(I3)')INT2
        IF ( INT2 .LT. 1.6*INT1 ) GO TO 250
        CALL ERRLOG ('ERROR015')
250 CONTINUE
C
C* ERROR016
C
        IF ( P(CURRSTR(30:31)) .NE. '06' ) GO TO 260
        IF ( P(CURRSTR(39:40)) .EQ. '01' ) GO TO 260
        CALL ERRLOG ('ERROR016')
260 CONTINUE
C
C* ERROR017
C
        IF ( P(CURRSTR(30:31)) .NE. '18' ) GO TO 270
        IF ( P(CURRSTR(39:40)) .EQ. '02' ) GO TO 270
        CALL ERRLOG ('ERROR017')
270 CONTINUE
C
C* ERROR018
C
        IF (CURRSTR(29:29) .NE. '1' ) GO TO 280
        IF ((CURRSTR(23:23).EQ.' ') .AND. (CURRSTR(24:24).EQ.' ') .AND.
1      (CURRSTR(25:25).EQ.' ') .AND. (CURRSTR(26:26).EQ.' ') .AND.
2      (CURRSTR(27:27).EQ.' ')) GO TO 280
        CALL ERRLOG ('ERROR018')
280 CONTINUE
C
C* ERROR019
C
        IF (CURRSTR(30:31) .EQ. ' ' ) GO TO 290
        IF (( P(CURRSTR(30:31)).EQ.'06') .OR. ( P(CURRSTR(30:31)).EQ.'18'
1)) GO TO 290
        CALL ERRLOG ('ERROR019')
290 CONTINUE
C
C* ERROR020
C
        IF (CURRSTR(42:44) .EQ. '999') GO TO 300
        IF (CURRSTR(42:44) .LT. '500' ) GO TO 300
        CALL ERRLOG ('ERROR020')
300 CONTINUE
C
C* ERROR021

```

```

C
    IF (CURRSTR(45:47) .EQ. '999') GO TO 310
    IF (CURRSTR(45:47) .LT. '500' ) GO TO 310
    CALL ERRLOG ('ERROR021')
310 CONTINUE
C
C* ERROR022
C
    IF (CURRSTR(48:50) .EQ. '999') GO TO 320
    IF (CURRSTR(48:50) .LT. '500' ) GO TO 320
    CALL ERRLOG ('ERROR022')
320 CONTINUE
C
C* ERROR023
C
    IF (CURRSTR(54:56) .EQ. '999') GO TO 330
    IF (CURRSTR(54:56) .LT. '500' ) GO TO 330
    CALL ERRLOG ('ERROR023')
330 CONTINUE
C
C* ERROR024
C
    IF (CURRSTR(57:59) .EQ. '999') GO TO 340
    IF (CURRSTR(57:59) .LT. '500' ) GO TO 340
    CALL ERRLOG ('ERROR024')
340 CONTINUE
C
C* ERROR025
C
    IF (CURRSTR(60:62) .EQ. '999') GO TO 350
    IF (CURRSTR(60:62) .LT. '500') GO TO 350
    CALL ERRLOG ('ERROR025')
350 CONTINUE
C
C* ERROR026
C
    IF (CURRSTR(67:69) .EQ. '999') GO TO 360
    IF (CURRSTR(67:69) .LT. '500') GO TO 360
    CALL ERRLOG ('ERROR026')
360 CONTINUE
C
C* ERROR027
C
    IF (CURRSTR(70:72) .EQ. '999') GO TO 370
    IF (CURRSTR(70:72) .LT. '500') GO TO 370
    CALL ERRLOG ('ERROR027')
370 CONTINUE
C
C* EROOR028
C
    IF (CURRSTR(73:75) .EQ. '999') GO TO 380
    IF (CURRSTR(73:75) .LT. '500' ) GO TO 380
    CALL ERRLOG ('ERROR028')
380 CONTINUE
C* WRITE ANY RECORDS WHICH ARE IN ERROR

```



```

      CALL ERRCHK
C
C* SET CURRENT RECORD TO PREVIOUS RECORD AND READ IN NEXT RECORD
C
      PREVSTR(1:80)=CURRSTR(1:80)
    901 CONTINUE
      READ(10,9001,END=995)CURRSTR
    9001 FORMAT(A80)
      VAR1=CURRSTR(30:31)//CURRSTR(33:38)
      VAR2=PREVSTR(30:31)//PREVSTR(33:38)
      IF ((VAR1.EQ.VAR2).AND.(KOUNT.LT.10)) THEN
        KOUNT=KOUNT + 1
      ELSE
        CALL WRAPUP (STORE,KOUNT)
        KOUNT=1
        LOGGRP=LOGGRP + 1
C
C* DETERMINE THE NUMBER OF SIGNIFICANT CHARACTERS IN LOGGRP
C
      NUMST=FSTRX(LOGGRP)
      STARTC=FBRKCX(NUMST,'123456789')
      IF (STARTC .EQ. 0) STARTC=1
C
C* HEAD UP OUTPUT PAGE
C
      HEAD='*** TRSPTEDT---INPUT FILE: '//INFILE(1:12)//
    1  ' LOGICAL GROUP '//NUMST(STARTC:)
      CALL EJECTH(HEAD)
      CALL EJECTB(54)
      END IF
      STORE(KOUNT)=CURRSTR
      GO TO 50
    995 CALL WRAPUP (STORE,KOUNT)
    999 CALL CLLGCT ('ECFSTR. ')
      STOP
      END

```

APPENDIX 3

User's Guide for the Tooth Reading Edit Process

USER'S GUIDE FOR THE TOOTH READING RECORD DATA BASE EDIT PROCESS

INCLUDED WITH THIS DOCUMENTATION IS A SCHEMATIC DIAGRAM OF THE TOOTH READING EDIT SYSTEM. THE DIAGRAM CAN BE DIVIDED INTO 3 PARTS:

- A. THE TOOTH READING EDIT PROCESS WHERE THE UNEDITED DATA FILE IS RUN THROUGH THE EDIT PROGRAM (TRSPTEDT) TO EDIT THE FILE
- B. SPLITTING THE EDITED DATA FILE INTO A FILE OF GOOD DATA (.AOK) AND BAD DATA (.BAD) USING THE PROGRAM GOODBAD.
- C. REFORMATING AND INDEXING THE FILE CONTAINING GOOD DATA BY USING THE PROGRAM REFORMAT.

THE TEXT BELOW DISCUSSES EACH PART IN DETAIL AND DESCRIBES THE NAMING CONVENTION FOR TOOTH READING DATA FILES.

NAMING OF DATA FILES:

DATA FILES ARE NAMED USING 8 ALPHA CHARACTERS FOLLOWED BY A FILE TYPE SUCH AS .DAT AND .ALL .

WORKING FILES:

BEGIN WITH (CHAR. 1-2) XX
CHAR. 3-4 ARE FOR TOOTH READING TR
CHAR. 5 IS FOR SEX M OR F
CHAR. 6-8 ARE FOR STOCK (SPOTTER)... SPT
(OR SPINNER=SPN)

FILE TYPE DESIGNATES THE TYPE OF RECORD:

.ALL = TOTAL DATA FILE
.AOK = SUBFILE OF .ALL FOR RECORDS TO BE USED
IN ANALYSES.
.BAD = SUBFILE OF .ALL OF "BAD" RECORDS, E.G.,
SPECIMEN MIX-UPS.
.DAT = COMPLETED, REFORMATTED, INDEXED .AOK FILE.

COMPLETED FILES:

WHEN A WORKING FILE IS COMPLETE AND READY FOR ANALYSIS,
THE FILE IS REFORMATTED AND INDEXED (SEE C.2). THIS
MAKES IT COMPATIBLE WITH OTHER DOPHIN DATA BASES.

A. TOOTH READING EDIT PROGRAM - **TRSPTEDT**

1. PREPARATION - THE TOOTH READING DATAFILE MUST EXIST ON DISK.
THE NAME FOR THE FILE SHOULD START WITH 'XXTR' AND END WITH '.ALL'.
FOR EXAMPLE, THE DATA FOR FEMALE SPOTTERS WOULD BE CALLED XXTRFSPT.ALL
THE FILE MUST BE SORTED BY READER CODE, READING DAY, AND READING OF DAY.
(THE USER CAN USE THE SORT COMMAND: \$SORT/KEY=(POS:30,SIZE:2)
/KEY=(POS:33,SIZE:6)/KEY=(POS:1,SIZE:3) INFILE OUTFILE).
A FILE CALLED ECFSTR. MUST ALSO EXIST ON DISK (IT CAN BE EMPTY).
2. BASIC PROCESSING LOGIC - THE PROGRAM FIRST GOES THROUGH A PRE-EDIT ROUTINE TO MAKE SURE THE CARDS ARE PROPERLY SORTED. IF ANY CARDS ARE OUT OF SEQUENCE, THEY WILL BE PRINTED OUT. PROCESSING WILL CONTINUE UNTIL ALL CARDS ARE CHECKED, BUT WILL NOT PROCEED ON TO THE MAIN PART OF THE EDIT PROGRAM IF ANY CARDS ARE FOUND OUT OF SEQUENCE. EXCEPTION: IF MORE THAN 25 CARDS ARE FOUND TO BE OUT OF SEQUENCE, PROCESSING WILL TERMINATE AFTER THE ERROR COUNT EXCEEDS 25. ONCE IN THE MAIN PART OF THE EDIT PROGRAM, CARDS ARE CHECKED FOR

ERRORS AND PRINTED OUT IF ERRORS ARE FOUND.

3. EXECUTING THE PROGRAM - THE PROGRAM CAN BE RUN BY GIVING THE COMMAND: \$SUBMIT TRSPTEDT. THE USER SHOULD MAKE SURE THE NAME OF THE DATAFILE IN THE PROGRAM TRSPTEDT.COM IS THE ONE S/HE PLANS TO EDIT. TO CHANGE THE NAME OF THIS DATAFILE GET INTO THE EDITOR WITH THE COMMAND: \$EDIT TRSPTEDT.COM. CHANGE THE FOURTH LINE TO THE NAME OF THE DATAFILE THAT IS TO BE EDITED.
 4. OUTPUT - THE PROGRAM WILL CREATE AN OUTPUT FILE ON DISK CALLED TRSPTEDT.LOG WHICH WILL SHOW THE EDIT ERRORS
 5. COMPILING AND LINKING - SOMETIMES THE PROGRAM MAY HAVE TO BE REVISED WHICH ALSO MEANS THAT THE PROGRAM HAS TO BE RECOMPILED AND RELINKED. THE PROGRAM CAN BE COMPILED WITH THE COMMAND: \$FOR TRSPTEDT. IT CAN BE LINKED WITH THE FOLLOWING COMMAND:
\$LINK TRSPTEDT,CSOOTR,UDISK4:[NM32.COMMONOBJ]BLANKT,CSEQ,EJECT,ERRCHK,ERRLOG,GENER1,GENER2,GENER3,P,PGROUP,RANGIT,VALUIT,VERFIT,WRAPUP
 6. A LIST OF THE INTERVARIABLE ERROR CHECKS EXISTS ON DISK WITH THE NAME TRSPTES.LIS.
 7. THE TOOTH READING EDIT ERROR CHECKS APPLY ONLY TO SPOTTERS. SOME DELIMITERS WILL CHANGE FOR OTHER STOCKS (E.G., STOCK CODE, READER CODES, RANGES FOR PREPARATION DATE AND READING DATE).
- B. SPLITTING THE DATAFILE INTO GOOD AND BAD DATA - **GOODBAD**
1. PREPARATION - AFTER EDITING THE FILE AND MAKING CORRECTIONS, THE USER SHOULD INDICATE WHICH DATA RECORDS WILL REMAIN BAD OR UNCORRECTED BY PUTTING AN ASTERISK IN COL.80.
 2. BASIC PROCESSING LOGIC - THE GOODBAD PROGRAM WILL CREATE TWO NEW FILES FROM THE .ALL DATA FILE, ONE WITH A FILE TYPE OF .AOK FOR THE GOOD DATA (E.G., XXTRFSPT.AOK) AND THE OTHER WITH A FILE TYPE OF .BAD (E.G., XXTRFSPT.BAD).
 3. EXECUTING THE PROGRAM - TO RUN THE PROGRAM, GIVE THE COMMAND \$RUN GOODBAD. THE PROGRAM WILL PROMPT YOU FOR THE FIRST 8 CHARACTERS OF THE INPUT FILE, FOR EXAMPLE 'XXTRFSPT'.
 4. COMPILING AND LINKING - TO COMPILE THE PROGRAM USE THE COMMAND \$FOR GOODBAD. TO LINK THE PROGRAM GIVE THE COMMAND \$LINK GOODBAD.
- C. REFORMATING AND INDEXING THE EDITED FILE - **REFORMAT**
1. PREPARATION - THE EDITED TOOTH READING FILE WILL HAVE A FILE TYPE OF .AOK. TO MAKE THIS FILE INTO AN INDEXED TOOTH READING DATABASE, THE REFORMAT PROGRAM MUST BE USED.
 2. BASIC PROCESSING LOGIC - THIS PROGRAM REFORMATS AND INDEXES AN .AOK DATA FILE (E.G., XXTRFSPT.AOK) FOR CONSISTENCY AND COMPATABILITY WITH OTHER ESTABLISHED DATA BASES. TWO DIFFERENT KEYS WERE CREATED FOR THE INDEXED FILE: THE PRIMARY (UNIQUE) KEY OF READER, READING DATE, READING OF DAY, AND THE SECONDARY (NON-UNIQUE) KEY OF SPECIMEN NUMBER. BECAUSE THE

PRIMARY KEY ELEMENTS MUST BE GROUPED TOGETHER, THE DATA FILE IS REFORMATTED TO PLACE THESE THREE ELEMENTS IN THE FIRST ELEVEN COLUMNS, FOLLOWED BY THE SPECIMEN NUMBER. AFTER COLUMN 38, ALL OTHER ELEMENTS REMAIN IN THE ORIGINAL FORMAT OF THE TOOTH READING RECORD CODE FORMS. THE REFORMAT PROGRAM ALSO CHANGES THE FILE NAME BY STRIPPING AWAY THE 'XX' AT THE FRONT, ADDING 'DB' AT THE END OF THE FILENAME, AND CHANGING THE FILE TYPE TO .DAT (E.G., TRFSPTDB.DAT).

3. EXECUTING THE PROGRAM - TO RUN THE PROGRAM GIVE THE COMMAND
\$RUN REFORMAT. THE PROGRAM WILL PROMPT YOU FOR THE NAME OF THE INPUT FILE. MAKE SURE THE NAME BEGINS WITH 'XX', FOR EXAMPLE, XXTRFSPT.AOK.
4. COMPILING AND LINKING - TO COMPILE THE PROGRAM USE THE COMMAND
\$FOR REFORMAT. TO LINK THE PROGRAM, GIVE THE COMMAND \$LINK
REFORMAT,UDISK4:[NM32.COMMONOBJ]FBRKCS

APPENDIX 4

VAX Fortran Data File Format Programs

APPENDIX 4A

```

C*****
C
C* PROGRAM:  GOODBAD
C
C* PURPOSE:  TO SPLIT A TOOTH READING DATAFILE INTO TWO FILES,
C             ONE OF 'GOOD' DATA AND ONE OF 'BAD' DATA.
C             BAD DATA IS DEFINED AS HAVING AN ASTERISK IN COL. 80.
C
C* PROGRAMMER:  G. JAY WALKER, MAY 1982, VAX FORTRAN
C
C*****
      CHARACTER NAME*8,XNAME*12,YNAME*12,ZNAME*12,REC*83
50  PRINT *, 'READ IN THE DATABASE NAME - FIRST 8 CHARACTERS'
      READ(5,9001) NAME
9001  FORMAT(A8)
      XNAME=NAME//'.ALL'
      YNAME=NAME//'.AOK'
      ZNAME=NAME//'.BAD'
      OPEN (UNIT=10,NAME=XNAME,STATUS='OLD',CARRIAGECONTROL='LIST')
      OPEN (UNIT=20,NAME=YNAME,STATUS='NEW',CARRIAGECONTROL='LIST')
      OPEN (UNIT=30,NAME=ZNAME,STATUS='NEW',CARRIAGECONTROL='LIST')
100  READ(10,9003,ERR=995,END=999) REC
9003  FORMAT(A83)
      IF (REC(80:80) .NE. '*') THEN
          WRITE(20,9003) REC
      ELSE
          WRITE(30,9003) REC
      END IF
      GO TO 100
995  PRINT *, 'FILE NOT ON DIRECTORY'
999  STOP
      END

```

APPENDIX 4B

```

C*****
C
C* PROGRAM:  LASTDATE
C
C* PURPOSE:  TO FIND THE LAST READING DATE FOR EACH SPECIMEN
C             WITHIN READING CODE OF A TOOTH READING DATA FILE
C             AND WRITE THE RECORDS TO AN OUTPUT FILE.
C
C* PROGRAMMER:  G. JAY WALKER, JUNE 1982, VAX FORTRAN
C
C*****
      CHARACTER REC*80,PRREC*80,KEY*9,PRKEY*9
      OPEN (UNIT=10,NAME='XXTRMSPT.AOK',STATUS='OLD',CARRIAGECONTROL=
1 'LIST')
      OPEN (UNIT=20,NAME='MSPTLAST.DAT',STATUS='NEW',CARRIAGECONTROL=
1 'LIST')
      READ(10,9001) REC
9001  FORMAT(A80)
      PRKEY=REC(4:10)//REC(30:31)
      PRREC=REC
50  READ(10,9001,END=999) REC
      KEY=REC(4:10)//REC(30:31)
      IF (KEY .NE. PRKEY) THEN
      WRITE(20,9001) PRREC
      END IF
      PRKEY=KEY
      PRREC=REC
      GO TO 50
999  WRITE(20,9001) PRREC
      CLOSE (UNIT=10)
      CLOSE (UNIT=20)
      STOP
      END

```


APPENDIX 4C

```

C*****
C
C* PROGRAM: REFORMAT
C
C* PURPOSE: TO REFORMAT TOOTH READING DATA FILES AND MAKE THEM INDEXED FILES
C
C* PROGRAMMER: G JAY WALKER , JUNE 1982, VAX FORTRAN
C*****
C
      CHARACTER INFILE*20,OUTFILE*20,REC*80,NEWREC*80,STRING*1
      INTEGER FBRKCS
      DATA STRING /'.'/
      20 PRINT *, 'ENTER THE NAME OF THE FILE TO BE REFORMATED AND INDEXED'
      READ(5,9001) INFILE
      9001 FORMAT(A20)
C* MAKE SURE THE INPUT FILE NAME STARTS WITH 'XX'.
      IF (INFILE(1:2) .NE. 'XX') GO TO 990
C* CHECK TO SEE THAT THERE IS A PERIOD IN THE FILE NAME
      ITEST=FBRKCS(INFILE,STRING)
      IF (ITEST .EQ. 0) GO TO 990
      KVAL=ITEST - 1
C
C* FOR THE OUTPUT FILE, STRIP OFF THE 'XX' AT THE BEGINNING OF THE NAME
C* AND ADD 'DB.DAT' AT THE END
C
      OUTFILE=INFILE(3:KVAL)//'DB.DAT'
C
C* OPEN THE INPUT FILE (SEQUENTIAL) AND THE OUTPUT FILE (INDEXED)
C
      OPEN (UNIT=10,NAME=INFILE,STATUS='OLD',CARRIAGECONTROL='LIST')
      OPEN (UNIT=20,NAME=OUTFILE,STATUS='NEW',CARRIAGECONTROL='LIST',
      1FORM='FORMATTED',ORGANIZATION='INDEXED',ACCESS='KEYED',KEY=
      2(1:11:CHARACTER),RECORDTYPE='FIXED',RECORDSIZE=80)
C
C* READ A RECORD FROM THE INPUT FILE
C
      50 READ(10,9002,END=999) REC
      9002 FORMAT(A80)
C
C* REFORMAT THE RECORD
C
      NEWREC(1:2)=REC(30:31)
      NEWREC(3:8)=REC(33:38)
      NEWREC(9:11)=REC(1:3)
      NEWREC(12:18)=REC(4:10)
      NEWREC(19:37)=REC(11:29)
      NEWREC(38:38)=REC(32:32)
      NEWREC(39:80)=REC(39:80)
C
C* WRITE THE REFORMATED RECORD TO THE OUTPUT FILE
C
      WRITE(20,9002) NEWREC
      GO TO 50
      990 PRINT *, 'INVALID ENTRY - TRY AGAIN'

```

```
      GO TO 20
999  CLOSE (UNIT=10)
      CLOSE (UNIT=20)
      STOP
      END
```

APPENDIX 4D

```

C*****
C
C* FILE: INDEX
C
C* PURPOSE: TO CHANGE A FILE OF 80-CHARACTER RECORDS FROM SEQUENTIAL TO INDEXED
C
C* PROGRAMMER: G JAY WALKER, OCTOBER 1982, VAX FORTRAN
C
C*****
C
C* TO COMPILE AND LINK THE PROGRAM GIVE THE COMMANDS:
C*   $FOR INDEX
C*   $LINK INDEX
C
C* TO RUN THE PROGRAM GIVE THE COMMANDS:
C*   $RUN INDEX
C*   FILENAME           (WHERE FILENAME IS THE FILE TO BE CONVERTED TO
C*                       AN INDEXED FILE)
C
C*****
C
CHARACTER REC*80,NEWREC*80,FILE*40
PRINT *, 'ENTER THE NAME OF THE FILE TO BE INDEXED '
READ(5,9001) FILE
9001 FORMAT(A40)
C
C* OPEN THE INPUT FILE IS A SEQUENTIAL FILE AND THE OUTPUT FILE AS AN
C* INDEXED FILE
C
OPEN (UNIT=10,NAME=FILE,STATUS='OLD',CARRIAGECONTROL=
1'LIST')
OPEN (UNIT=20,NAME=FILE,STATUS='NEW',CARRIAGECONTROL=
1'LIST',FORM='FORMATTED',ORGANIZATION='INDEXED',ACCESS='KEYED',
2KEY=(1:7:CHARACTER),RECORDTYPE='FIXED',RECORDSIZE=80)
C
C* READ A RECORD FROM THE INPUT FILE (SEQUENTIAL)
C
50 READ(10,9003,END=999)REC
9003 FORMAT(A80)
NEWREC(1:80)=REC(1:80)
C
C* WRITE A RECORD TO THE OUTPUT FILE (INDEXED)
C
WRITE(20,9002)NEWREC
9002 FORMAT(A80)
GO TO 50
999 CLOSE (UNIT=10)
CLOSE (UNIT=20)
STOP
END

```

APPENDIX 4E

```

C*****
C
C* PROGRAM: MALE
C
C* PURPOSE: TO PUT A '1' IN COL. 79 OF THE MALE SPOTTER TOOTH
C            READING DATA FILE (XXTRMSPT.AOK) TO DISTINGUISH IT
C            FROM FEMALE DATA RECORDS. AFTER RUNNING 'MALE' THE
C            FEMALE AND MALE SPOTTERS CAN BE MERGED FOR ANALYSES.
C            TO SEPARATE THE FILES AGAIN USE MFSPLIT.FOR .
C
C* PROGRAMMER: G. JAY WALKER, 'SEPTEMBER 1982, VAX FORTRAN
C
C*****
      CHARACTER REC*83, INFILE*20, OUTFILE*20
      PRINT *, 'ENTER INPUT FILE NAME'
      READ (5,9003) INFILE
      OPEN (UNIT=10,FILE=INFILE,STATUS='OLD',CARRIAGECONTROL
+   ='LIST')
      PRINT *, 'ENTER OUTPUT FILE NAME'
      READ (5,9003) OUTFILE
      OPEN (UNIT=20,FILE=OUTFILE,STATUS='NEW',CARRIAGECONTROL
+   ='LIST')
100  READ(10,9003,END=999) REC
9003  FORMAT(A83)
      REC (79:79) = '1'
      WRITE(20,9003) REC
      GO TO 100
999  STOP
      END

```


APPENDIX 4F

```

C*****
C
C* PROGRAM:  MFSPLIT
C
C* PURPOSE:  TO SPLIT A TOOTH READING DATAFILE INTO TWO FILES,
C             ONE OF 'MALE' DATA AND ONE OF 'FEMALE' DATA.
C             MALE DATA IS RECOGNIZED BY A '1' IN COL. 79.
C
C* PROGRAMMER:  G. JAY WALKER, SEPTEMBER 1982, VAX FORTRAN
C
C*****
      CHARACTER NAME*8,XNAME*12,YNAME*12,ZNAME*12,REC*83
50  PRINT *, 'READ IN THE DATABASE NAME - FIRST 8 CHARACTERS'
      READ(5,9001) NAME
9001  FORMAT(A8)
      XNAME=NAME//'.ALL'
      YNAME=NAME//'.FEM'
      ZNAME=NAME//'.MAL'
      OPEN (UNIT=10,NAME=XNAME,STATUS='OLD',CARRIAGECONTROL='LIST')
      OPEN (UNIT=20,NAME=YNAME,STATUS='NEW',CARRIAGECONTROL='LIST')
      OPEN (UNIT=30,NAME=ZNAME,STATUS='NEW',CARRIAGECONTROL='LIST')
100  READ(10,9003,ERR=995,END=999) REC
9003  FORMAT(A83)
      IF (REC(79:79) .NE. '1') THEN
          WRITE(20,9003) REC
      ELSE
          WRITE(30,9003) REC
      END IF
      GO TO 100
995  PRINT *, 'FILE NOT ON DIRECTORY'
999  STOP
      END

```

APPENDIX 5

VAX Fortran Data Extraction Programs

APPENDIX 5A

```

C *****
C
C* PROGRAM:  MAGEBST.FOR
C
C* PURPOSE:  TO CALCULATE MEANS, VARIANCES, T VALUES, POOLED
C             MEAN, AND POOLED VARIANCES OF 'AGEBEST' TOOTH
C             READINGS FOR EACH SPECIMEN BY READER.
C
C* PROGRAMMER:  SUSAN BOYER, AUGUST 1982, VAX FORTRAN
C
C *****
C             CHARACTER STRING1*41,RDR*2,SPEC*7,INFILE*20,OUTFILE*20,
C             RDR1*2,RDR2*2,STRING2*38
C             INTEGER IER
C             REAL TVAL,DF,Q,N1,N2,NP
C
C ENTER FILE NAMES
C
C             PRINT *,'---ENTER INPUT FILE NAME'
C             READ (5,9000)INFILE
C
C             PRINT *,'---ENTER OUTPUT FILE NAME'
C             READ (5,9000)OUTFILE
C
C OPEN FILES
C
C             OPEN(UNIT=10,FILE=INFILE,STATUS='OLD')
C             OPEN(UNIT=20,FILE=OUTFILE,STATUS='NEW',CARRIAGECONTROL='LIST')
C
C READ FIRST RECORD, INITIALIZE RECORD COUNTER
C
C             READ(10,9000,END=999)STRING1,AGE,STRING2,CYCLE
C             NRECS=1
C             RDR1='0'
C             RDR2='0'
C             N1=0.0
C             N2=0.0
C
C BEGIN FILE LOOP
C
C 100  CONTINUE
C             SPEC=STRING1(4:10)
C             RDR=STRING1(30:31)
C             SX1=0
C             SX12=0
C             SX2=0
C             SX22=0
C
C BEGIN SPECIMEN LOOP
C
C 200  CONTINUE
C             IF(SPEC.NE.STRING1(4:10)) THEN
C
C COMPUTE MEANS, VARIANCES, T, EXACT PROBABILITY OF T
C CHECK FOR N'S TOO SMALL FOR VARIANCE CALC.
C

```

```

IF(N1.LE.1.0)GOTO 400
IF(N2.LE.1.0)GOTO 500
C
SXP=SX1+SX2
SXP2=SX12+SX22
NP=N1+N2
XB1=SX1/N1
XB2=SX2/N2
XBP=(XB1+XB2)/2
SS1=SX12-((SX1*SX1)/N1)
SS2=SX22-((SX2*SX2)/N2)
V1=SS1/(N1-1)
V2=SS2/(N2-1)
VP1=(SS1+SS2)/(NP-2)
VP2=(SXP2-((SXP**2)/NP))/(NP-1)
IF(V1.EQ.0.0.OR.V2.EQ.0.0)THEN
    T=99.9
    PT=0.00
    GOTO 250
ELSE
    T=(XB1-XB2)/((VP1*(NP/(N1*N2))))**0.5)
    CALL MDTD(T,(NP-2),Q,IER)
    PT=Q
ENDIF
250 CONTINUE
WRITE(20,9009)SPEC, RDR1,XB1,V1,N1,RDR2,XB2,V2,
    N2,XBP,VP1,VP2,T,PT
N1=0
N2=0
GOTO 100

ELSEIF (RDR.NE.STRING1(30:31))THEN
C
C SUMS OF SQUARES FOR READER 2
C CHECK FOR UNRECORDED AGE (99.9)
C
    IF(AGE.EQ.99.9)GOTO 300
C
    RDR1=RDR
    RDR2=STRING1(30:31)
    SX2=SX2+AGE
    SX22=SX22+(AGE**2)
    N2=N2+1
    GOTO 300
ELSE
C
C SUMS OF SQUARES FOR READER 1
C CHECK FOR UNRECORDED AGE (99.9)
C
    IF(AGE.EQ.99.9)GOTO 300
C
    SX1=SX1+AGE
    SX12=SX12+(AGE**2)
    N1=N1+1
ENDIF

```



```

C
C  READ NEW RECORD
C
300 READ(10,9000,END=999)STRING1,AGE,STRING2,CYCLE
    NRECS=NRECS+1
    GOTO 200
C
400  IF (N1 .EQ. 0.0) SX1=99.9
    XB1=SX1
    IF (N2 .EQ. 0.0) SX2=99.9
    V1=99.9
C
C  CHECK FOR N2<2
C
    IF(N2.LE.1.0)THEN
        XB2=SX2
        V2=99.9
        GOTO 450
    ELSE
        XB2=SX2/N2
        SS2=SX22-((SX2**2)/N2)
        V2=SS2/(N2-1)
    ENDIF
450  CONTINUE
    VP1=99.9
    VP2=99.9
    XBP=(XB1 + XB2) / 2.0
    IF ((XB1 .EQ. 99.9).AND. (XB2 .NE. 99.9)) XBP=XB2
    IF ((XB1 .NE. 99.9) .AND. (XB2 .EQ. 99.9)) XBP=XB1
    T=99.9
    PT=9.99
    GOTO 250
C
C
500  IF (N2 .EQ. 0.0) SX2 = 99.9
    XB2=SX2
    IF (N1 .EQ. 0.0) SX1 = 99.9
    V2=99.9
C
C  CHECK FOR N1<2
C
    IF(N1.LE.1.0)THEN
        XB1=SX1
        V1=99.9
        GOTO 550
    ELSE
        XB1=SX1/N1
        SS1=SX12-((SX1**2)/N1)
        V1=SS1/(N1-1)
    ENDIF
550  CONTINUE
    VP1=99.9
    VP2=99.9
    XBP = (XB1 + XB2) / 2.0
    IF ((XB1 .EQ. 99.9).AND.(XB2.NE.99.9)) XBP=XB2

```

```
      IF ((XB1 .NE. 99.9).AND.(XB2.EQ.99.9)) XBP=XB1
      T=99.9
      PT=9.99
      GOTO 250
999   continue
      PRINT *, '-----NUMBER OF RECORDS PROCESSED = ', NRECS
9000  FORMAT(A,F3.1,A,F2.0)
9009  FORMAT(1X,A7,2X,2(A2,2X,2(F8.2,2X),F3.0,2X),4(F8.2,2X),F6.4)
      END
```

APPENDIX 5B

```

C*****
C
C* PROGRAM: MAKEDAT
C
C* PURPOSE: TO EXTRACT INFORMATION FROM THE FEMALE SPOTTER LIFE HISTORY
C            DATABASE, THE AGEBESTF DATAFILE, AND THE FSPTLAST FILE AND
C            WRITE IT TO A NEW FILE CALLED FLHAGE.DAT .
C
C* PROGRAMMER: G. JAY WALKER, AUGUST 1982, VAX FORTRAN
C
C*****
C
C* COMPILING AND LINKING - USE THE COMMANDS:
C*      $FOR MAKEDAT
C*      $LINK MAKEDAT
C
C* EXECUTION - USE THE COMMAND:
C*      $RUN MAKEDAT
C
C      CHARACTER*3 BAGE1,HAGE1,LAGE1,BAGE2,HAGE2,LAGE2
C      CHARACTER*5 MEAN06,MEAN18,OVERM
C      CHARACTER*7 SPEC
C      CHARACTER*8 M06,M18,OM
C      CHARACTER*151 LHREC
C
C* OPEN THE INPUT AND OUTPUT FILES
C
C      OPEN (UNIT=10,NAME='[NM27.SPTEDT]FSPTLAST.DAT',STATUS='OLD',
C1CARRIAGECONTROL='LIST')
C      OPEN (UNIT=20,NAME='[NM27.ALETA]AGEBESTF.DAT',STATUS='OLD',
C1CARRIAGECONTROL='LIST',FORM='FORMATTED',ORGANIZATION='INDEXED',
C2ACCESS='KEYED',KEY=(2:8:CHARACTER),RECORDTYPE='FIXED',
C3RECORDSIZE=114)
C      OPEN (UNIT=30,NAME='[NM27.SPTEDT]FSPTDB.DAT',STATUS='OLD',
C1CARRIAGECONTROL='LIST',FORM='FORMATTED',ORGANIZATION='INDEXED',
C2ACCESS='KEYED',KEY=(1:7:CHARACTER),RECORDTYPE='FIXED',
C3RECORDSIZE=151)
C      OPEN (UNIT=40,NAME='FLHAGE.DAT',STATUS='NEW',CARRIAGECONTROL=
C1'LIST',RECORDSIZE=184)
C
C* READ A RECORD FROM THE FSPTLAST FILE
C
C      50 READ(10,9001,END=990) SPEC,IRCODE,BAGE1,HAGE1,LAGE1
C      9001 FORMAT(3X,A7,19X,I2,10X,3A3)
C            IF (IRCODE .NE. 6) GO TO 960
C            READ(10,9001,END=990) SPEC, IRCODE, BAGE2, HAGE2, LAGE2
C            IF (IRCODE .NE. 18) GO TO 960
C
C* LOOK FOR MATCHES ON THE AGEBESTF AND FSPTDB FILES
C
C      READ(20,9003,KEYEQ=SPEC,ERR=940) M06,M18,OM
C      9003 FORMAT(14X,A8,21X,A8,17X,A8)
C            MEAN06=M06(3:5)//M06(7:8)
C            MEAN18=M18(3:5)//M18(7:8)
C            OVERM=OM(3:5)//OM(7:8)

```

```

      READ(30,9005,KEYEQ=SPEC,ERR=950) LHREC
9005 FORMAT(A151)
C
C* WRITE OUT THE OUTPUT RECORD
C
      WRITE(40,9007) LHREC,OVERM,MEAN06,BAGE1,HAGE1,LAGE1,MEAN18,
1BAGE2,HAGE2,LAGE2
9007 FORMAT(A151,2A5,3A3,A5,3A3)
      GO TO 50
C
C* FOR UNMATCHED SPECIMEN NUMBERS
C
940 PRINT *, 'NO MATCH ON AGEBESTF FOR SPECIMEN= ',SPEC
      GO TO 50
950 PRINT *, 'NO MATCH ON FSPTDB FOR SPECIMEN= ',SPEC
      GO TO 50
C
C* FOR CARDS OUT OF SEQUENCE
C
960 PRINT *, 'READER CODE OUT OF ORDER FOR SPECIMEN = ',SPEC, ' - PROCES
1SING TERMINATES '
C
C* CLOSE THE DATABASES
C
990 CLOSE (UNIT=10)
      CLOSE (UNIT=20)
      CLOSE (UNIT=30)
      CLOSE (UNIT=40)
      STOP
      END

```


APPENDIX 5C

```

C*****
C
C* PROGRAM: MAKEDAT2
C
C* PURPOSE: TO EXTRACT INFORMATION FROM THE LHAGE FILES AND SET LOG
C           DATABASE INTO A NEW FILE CALLED SETAGE.DAT .
C
C* PROGRAMMER: G. JAY WALKER, JULY 1982, VAX FORTRAN
C
C*****
C
C* THE MAKEDAT2 PROGRAM EXTRACTS INFORMATION FROM THE LHAGE FILE (CREATED
C* BY MAKEDAT) AND THE SET LOG DATABASE. THE INFORMATION IS PUT INTO
C* AN OUTPUT FILE CALLED SETAGE. THE USER IS PROMPTED FOR THE YEAR THE
C* PROGRAM SHOULD BE RUN FOR. THE SET LOG FOR THAT YEAR SHOULD EXIST ON DISK.
C
C* THE OBSERVER CODES DO NOT EXIST ON THE SET LOG DATABASES FOR 1973 TO 1977.
C* THE VALUES FOR THOSE YEARS WILL BE SET TO BLANKS
C
C* TO COMPILE AND LINK THE PROGRAMS USE THE COMMANDS:
C*   $FOR MAKEDAT2
C*   $LINK MAKEDAT2,UDISK4:[NM32.COMMONOBJ]KGTEC,CHGETV,LNAME,LNUMB,FIVALS,
C*   FBRKCS,FVRFYS
C
C* TO RUN THE PROGRAM USE THE COMMAND:
C*   $RUN MAKEDAT2
C*   78 (WHERE 78 IS AN EXAMPLE OF THE YEAR THE PROGRAM IS TO BE RUN FOR)
C
C*****
C
CHARACTER*1 NS,EW
CHARACTER*2 YEAR,MONTH,DAY,LATD,LATM,LONGM,EYEAR
CHARACTER*3 CRUISE,SET,LONGD,E1SPT,E1SPN,E1OTH1,E1OTH2,
1          E2SPT,E2SPN,E2OTH1,E2OTH2,E3SPT,E3SPN,E3OTH1,E3OTH2,
2          OSPTB,OESTB,OWBB,OSPN2B,OOTH1B,OOTH2B,E1SPTC,E1SPNC,
3          E1OTH1C,E1OTH2C,E2SPTC,E2SPNC,E2OTH1C,E2OTH2C,
4          E3SPTC,E3SPNC,E3OTH1C,E3OTH2C,OSPTC,OESTC,OWBC,
5          OSPN2C,OOTH1C,OOTH2C,TGSPTK,TGSPNK,TGOTHK,TRSPTK,
6          TRSPNK,TROTHK,SKSPTK,SKSPNK,SKOTHK,OTSPTK,OTSPNK,
7          OTOTHK,UKSPTK,UKSPNK,UKOTHK
CHARACTER*4 E1TOT,E2TOT,E3TOT,OBSTB,OHIB,LOB,E1TOTC,E2TOTC,
1          E3TOTC,OBSTC,OHIC,LOC,KKILL
CHARACTER*6 KEYVAL
CHARACTER*10 FILE1
CHARACTER*31 KA(75)
CHARACTER*184 LHAGEREC
CHARACTER*262 NEWREC
CHARACTER*673 SLREC
C
C* PROMPT USER FOR THE YEAR THE PROGRAM IS TO BE RUN FOR
C
20 PRINT *, 'ENTER THE YEAR FOR WHICH YOU WANT TO RUN THE PROGRAM -
173 TO 78 '
READ(5,9001,ERR=40) EYEAR
9001 FORMAT(A2)

```

```

      IF ( (EYEAR .GT. '72') .AND. (EYEAR .LT. '79') ) GO TO 50
40 PRINT *, 'INVALID VALUE ENTERED - TRY AGAIN '
      GO TO 20
50 ISL=376
      IF (EYEAR .EQ. '76') ISL=450
      IF (EYEAR .GT. '76') ISL=673
      FILE1='SL'//EYEAR//'DB.DAT'

```

C

C* OPEN THE INPUT AND OUTPUT FILES

C

```

      OPEN (UNIT=10, NAME='FLHAGE.DAT', STATUS='OLD', CARRIAGECONTROL=
1 'LIST', RECORDSIZE=184)
      OPEN (UNIT=20, NAME=FILE1, STATUS='OLD', CARRIAGECONTROL
1='LIST', FORM='FORMATTED', ORGANIZATION='INDEXED', ACCESS='KEYED',
2KEY=(1:6:CHARACTER), RECORDTYPE='FIXED')
      OPEN (UNIT=30, NAME='SETAGE'//EYEAR//'.DAT', STATUS='OLD',
1CARRIAGECONTROL='LIST', RECORDSIZE=262)

```

C

C* DEFINE DATA ELEMENTS

C

```

      CALL KGTEC (IR, FILE1, 'CRUISE', KA(1))
      CALL KGTEC (IR, FILE1, 'SET', KA(2))
      CALL KGTEC (IR, FILE1, 'YEAR', KA(3))
      CALL KGTEC (IR, FILE1, 'MONTH', KA(4))
      CALL KGTEC (IR, FILE1, 'DAY', KA(5))
      CALL KGTEC (IR, FILE1, 'LATD', KA(6))
      CALL KGTEC (IR, FILE1, 'LATM', KA(7))
      CALL KGTEC (IR, FILE1, 'N-OR-S', KA(8))
      CALL KGTEC (IR, FILE1, 'LONGD', KA(9))
      CALL KGTEC (IR, FILE1, 'LONGM', KA(10))
      CALL KGTEC (IR, FILE1, 'E-OR-W', KA(11))
      CALL KGTEC (IR, FILE1, 'E1-PORP-TOT', KA(12))
      CALL KGTEC (IR, FILE1, 'E1-SPT', KA(13))
      CALL KGTEC (IR, FILE1, 'E1-SPN', KA(14))
      CALL KGTEC (IR, FILE1, 'E1-OTHER-SP1', KA(15))
      CALL KGTEC (IR, FILE1, 'E1-OTHER-SP2', KA(16))
      CALL KGTEC (IR, FILE1, 'E2-PORP-TOT', KA(17))
      CALL KGTEC (IR, FILE1, 'E2-SPT', KA(18))
      CALL KGTEC (IR, FILE1, 'E2-SPN', KA(19))
      CALL KGTEC (IR, FILE1, 'E2-OTHER-SP1', KA(20))
      CALL KGTEC (IR, FILE1, 'E2-OTHER-SP2', KA(21))
      CALL KGTEC (IR, FILE1, 'E3-PORP-TOT', KA(22))
      CALL KGTEC (IR, FILE1, 'E3-SPT', KA(23))
      CALL KGTEC (IR, FILE1, 'E3-SPN', KA(24))
      CALL KGTEC (IR, FILE1, 'E3-OTHER-SP1', KA(25))
      CALL KGTEC (IR, FILE1, 'E3-OTHER-SP2', KA(26))
      CALL KGTEC (IR, FILE1, 'E1-TOT-CATCH', KA(27))
      CALL KGTEC (IR, FILE1, 'E1-SPT-CATCH', KA(28))
      CALL KGTEC (IR, FILE1, 'E1-SPN-CATCH', KA(29))
      CALL KGTEC (IR, FILE1, 'E1-OTHER1-CATCH', KA(30))
      CALL KGTEC (IR, FILE1, 'E1-OTHER2-CATCH', KA(31))
      CALL KGTEC (IR, FILE1, 'E2-TOT-CATCH', KA(32))
      CALL KGTEC (IR, FILE1, 'E2-SPT-CATCH', KA(33))
      CALL KGTEC (IR, FILE1, 'E2-SPN-CATCH', KA(34))
      CALL KGTEC (IR, FILE1, 'E2-OTHER1-CATCH', KA(35))

```

```

CALL KGTEC (IR,FILE1,'E2-OTHER2-CATCH',KA(36))
CALL KGTEC (IR,FILE1,'E3-TOT-CATCH',KA(37))
CALL KGTEC (IR,FILE1,'E3-SPT-CATCH',KA(38))
CALL KGTEC (IR,FILE1,'E3-SPN-CATCH',KA(39))
CALL KGTEC (IR,FILE1,'E3-OTHER1-CATCH',KA(40))
CALL KGTEC (IR,FILE1,'E3-OTHER2-CATCH',KA(41))
CALL KGTEC (IR,FILE1,'KNOWN-KILLED',KA(42))
CALL KGTEC (IR,FILE1,'TANGL-SPT-KILL',KA(43))
CALL KGTEC (IR,FILE1,'TANGL-SPN-KILL',KA(44))
CALL KGTEC (IR,FILE1,'TANGL-OTH-KILL',KA(45))
CALL KGTEC (IR,FILE1,'TRAP-SPT-KILL',KA(46))
CALL KGTEC (IR,FILE1,'TRAP-SPN-KILL',KA(47))
CALL KGTEC (IR,FILE1,'TRAP-OTH-KILL',KA(48))
CALL KGTEC (IR,FILE1,'SACKUP-SPT-KILL',KA(49))
CALL KGTEC (IR,FILE1,'SACKUP-SPN-KILL',KA(50))
CALL KGTEC (IR,FILE1,'SACKUP-OTH-KILL',KA(51))
CALL KGTEC (IR,FILE1,'OTHER-SPT-KILL',KA(52))
CALL KGTEC (IR,FILE1,'OTHER-SPN-KILL',KA(53))
CALL KGTEC (IR,FILE1,'OTHER-OTH-KILL',KA(54))
CALL KGTEC (IR,FILE1,'UNK-SPT-KILL',KA(55))
CALL KGTEC (IR,FILE1,'UNK-SPN-KILL',KA(56))
CALL KGTEC (IR,FILE1,'UNK-OTH-KILL',KA(57))

```

C

C* IF THE YEAR IS BEFORE 1977, SKIP THE OBSERVER CODES

C

```

IF (EYEAR .LT. '77') GO TO 100
CALL KGTEC (IR,FILE1,'OBS-BST-EST-BEF',KA(58))
CALL KGTEC (IR,FILE1,'OBS-HI-EST-BEF',KA(59))
CALL KGTEC (IR,FILE1,'OBS-LO-EST-BEF',KA(60))
CALL KGTEC (IR,FILE1,'OBS-SPT-BEF',KA(61))
CALL KGTEC (IR,FILE1,'OBS-EASTERN-BEF',KA(62))
CALL KGTEC (IR,FILE1,'OBS-WB-BEF',KA(63))
CALL KGTEC (IR,FILE1,'OBS-SPN2-BEF',KA(64))
CALL KGTEC (IR,FILE1,'OBS-OTHR1-BEF',KA(65))
CALL KGTEC (IR,FILE1,'OBS-OTHR2-BEF',KA(66))
CALL KGTEC (IR,FILE1,'OBS-BST-EST-CAP',KA(67))
CALL KGTEC (IR,FILE1,'OBS-HI-EST-CAP',KA(68))
CALL KGTEC (IR,FILE1,'OBS-LO-EST-CAP',KA(69))
CALL KGTEC (IR,FILE1,'OBS-SPT-CAP',KA(70))
CALL KGTEC (IR,FILE1,'OBS-EASTERN-CAP',KA(71))
CALL KGTEC (IR,FILE1,'OBS-WB-CAP',KA(72))
CALL KGTEC (IR,FILE1,'OBS-SPN2-CAP',KA(73))
CALL KGTEC (IR,FILE1,'OBS-OTHR1-CAP',KA(74))
CALL KGTEC (IR,FILE1,'OBS-OTHR2-CAP',KA(75))

```

C

C* READ A RECORD FROM THE LHAGE DATAFILE

C

```

100 READ(10,9003,END=995) LHAGEREC
9003 FORMAT(A184)

```

C

C* TEST FOR YEAR EQUALING THE USER-ENTERED YEAR

C

```

IF (LHAGEREC(19:20) .NE. EYEAR) GO TO 100

```

C

C* SEARCH THE SET LOG DATABASE FOR A MATCH ON CRUISE AND SET

```

C
    KEYVAL=LHAGEREC(9:14)
    READ(20,9005,KEYEQ=KEYVAL,ERR=990) SLREC
9005 FORMAT(A)
C
C* RESET OBSERVER CODES TO BLANKS
C
    OBSTB=' '
    OHIB=' '
    OLOB=' '
    OSPTB=' '
    OESTB=' '
    OWBB=' '
    OSPN2B=' '
    OOTH1B=' '
    OOTH2B=' '
    OBSTC=' '
    OHIC=' '
    OLOC=' '
    OSPTC=' '
    OESTC=' '
    OWBC=' '
    OSPN2C=' '
    OOTH1C=' '
    OOTH2C=' '
C
C* EXTRACT INFORMATION FROM THE SET LOG DATABASE
C
    CALL CHGETV (IR,SLREC,KA(1),CRUISE)
    CALL CHGETV (IR,SLREC,KA(2),SET)
    CALL CHGETV (IR,SLREC,KA(3),YEAR)
    CALL CHGETV (IR,SLREC,KA(4),MONTH)
    CALL CHGETV (IR,SLREC,KA(5),DAY)
    CALL CHGETV (IR,SLREC,KA(6),LATD)
    CALL CHGETV (IR,SLREC,KA(7),LATM)
    CALL CHGETV (IR,SLREC,KA(8),NS)
    CALL CHGETV (IR,SLREC,KA(9),LONGD)
    CALL CHGETV (IR,SLREC,KA(10),LONGM)
    CALL CHGETV (IR,SLREC,KA(11),EW)
    CALL CHGETV (IR,SLREC,KA(12),E1TOT)
    CALL CHGETV (IR,SLREC,KA(13),E1SPT)
    CALL CHGETV (IR,SLREC,KA(14),E1SPN)
    CALL CHGETV (IR,SLREC,KA(15),E1OTH1)
    CALL CHGETV (IR,SLREC,KA(16),E1OTH2)
    CALL CHGETV (IR,SLREC,KA(17),E2TOT)
    CALL CHGETV (IR,SLREC,KA(18),E2SPT)
    CALL CHGETV (IR,SLREC,KA(19),E2SPN)
    CALL CHGETV (IR,SLREC,KA(20),E2OTH1)
    CALL CHGETV (IR,SLREC,KA(21),E2OTH2)
    CALL CHGETV (IR,SLREC,KA(22),E3TOT)
    CALL CHGETV (IR,SLREC,KA(23),E3SPT)
    CALL CHGETV (IR,SLREC,KA(24),E3SPN)
    CALL CHGETV (IR,SLREC,KA(25),E3OTH1)
    CALL CHGETV (IR,SLREC,KA(26),E3OTH2)
    CALL CHGETV (IR,SLREC,KA(27),E1TOTC)

```



```

CALL CHGETV (IR,SLREC,KA(28),E1SPTC)
CALL CHGETV (IR,SLREC,KA(29),E1SPNC)
CALL CHGETV (IR,SLREC,KA(30),E1OTH1C)
CALL CHGETV (IR,SLREC,KA(31),E1OTH2C)
CALL CHGETV (IR,SLREC,KA(32),E2TOTC)
CALL CHGETV (IR,SLREC,KA(33),E2SPTC)
CALL CHGETV (IR,SLREC,KA(34),E2SPNC)
CALL CHGETV (IR,SLREC,KA(35),E2OTH1C)
CALL CHGETV (IR,SLREC,KA(36),E2OTH2C)
CALL CHGETV (IR,SLREC,KA(37),E3TOTC)
CALL CHGETV (IR,SLREC,KA(38),E3SPTC)
CALL CHGETV (IR,SLREC,KA(39),E3SPNC)
CALL CHGETV (IR,SLREC,KA(40),E3OTH1C)
CALL CHGETV (IR,SLREC,KA(41),E3OTH2C)
CALL CHGETV (IR,SLREC,KA(42),KKILL)
CALL CHGETV (IR,SLREC,KA(43),TGSPTK)
CALL CHGETV (IR,SLREC,KA(44),TGSPNK)
CALL CHGETV (IR,SLREC,KA(45),TGOTHK)
CALL CHGETV (IR,SLREC,KA(46),TRSPTK)
CALL CHGETV (IR,SLREC,KA(47),TRSPNK)
CALL CHGETV (IR,SLREC,KA(48),TROTHK)
CALL CHGETV (IR,SLREC,KA(49),SKSPTK)
CALL CHGETV (IR,SLREC,KA(50),SKSPNK)
CALL CHGETV (IR,SLREC,KA(51),SKOTHK)
CALL CHGETV (IR,SLREC,KA(52),OTSPTK)
CALL CHGETV (IR,SLREC,KA(53),OTSPNK)
CALL CHGETV (IR,SLREC,KA(54),OTOTHK)
CALL CHGETV (IR,SLREC,KA(55),UKSPTK)
CALL CHGETV (IR,SLREC,KA(56),UKSPNK)
CALL CHGETV (IR,SLREC,KA(57),UKOTHK)

```

C

C* IF THE YEAR IS BEFORE 1977, SKIP THE OBSERVER CODES

C

```

IF (EYEAR .LT. '77') GO TO 200
CALL CHGETV (IR,SLREC,KA(58),OBSTB)
CALL CHGETV (IR,SLREC,KA(59),OHIB)
CALL CHGETV (IR,SLREC,KA(60),OLOB)
CALL CHGETV (IR,SLREC,KA(61),OSPTB)
CALL CHGETV (IR,SLREC,KA(62),OESTB)
CALL CHGETV (IR,SLREC,KA(63),OWBB)
CALL CHGETV (IR,SLREC,KA(64),OSPN2B)
CALL CHGETV (IR,SLREC,KA(65),OOTH1B)
CALL CHGETV (IR,SLREC,KA(66),OOTH2B)
CALL CHGETV (IR,SLREC,KA(67),OBSTC)
CALL CHGETV (IR,SLREC,KA(68),OHIC)
CALL CHGETV (IR,SLREC,KA(69),OLOC)
CALL CHGETV (IR,SLREC,KA(70),OSPTC)
CALL CHGETV (IR,SLREC,KA(71),OESTC)
CALL CHGETV (IR,SLREC,KA(72),OWBC)
CALL CHGETV (IR,SLREC,KA(73),OSPN2C)
CALL CHGETV (IR,SLREC,KA(74),OOTH1C)
CALL CHGETV (IR,SLREC,KA(75),OOTH2C)

```

C

C* DEFINE ELEMENTS OF THE OUTPUT RECORD

C

```

200 NEWREC(1:7)=LHAGEREC(1:7)
    NEWREC(8:11)=LHAGEREC(37:40)
    NEWREC(12:16)=LHAGEREC(152:156)
    NEWREC(17:19)=LHAGEREC(162:164)
    NEWREC(20:22)=LHAGEREC(165:167)
    NEWREC(23:25)=LHAGEREC(168:170)
    NEWREC(26:28)=LHAGEREC(176:178)
    NEWREC(29:31)=LHAGEREC(179:181)
    NEWREC(32:34)=LHAGEREC(182:184)
    NEWREC(35:37)=CRUISE
    NEWREC(38:40)=SET
    NEWREC(41:42)=YEAR
    NEWREC(43:44)=MONTH
    NEWREC(45:46)=DAY
    NEWREC(47:48)=LATD
    NEWREC(49:50)=LATM
    NEWREC(51:51)=NS
    NEWREC(52:54)=LONGD
    NEWREC(55:56)=LONGM
    NEWREC(57:57)=EW
    NEWREC(58:61)=E1TOT
    NEWREC(62:64)=E1SPT
    NEWREC(65:67)=E1SPN
    NEWREC(68:70)=E1OTH1
    NEWREC(71:73)=E1OTH2
    NEWREC(74:77)=E2TOT
    NEWREC(78:80)=E2SPT
    NEWREC(81:83)=E2SPN
    NEWREC(84:86)=E2OTH1
    NEWREC(87:89)=E2OTH2
    NEWREC(90:93)=E3TOT
    NEWREC(94:96)=E3SPT
    NEWREC(97:99)=E3SPN
    NEWREC(100:102)=E3OTH1
    NEWREC(103:105)=E3OTH2
    NEWREC(106:109)=E1TOTC
    NEWREC(110:112)=E1SPTC
    NEWREC(113:115)=E1SPNC
    NEWREC(116:118)=E1OTH1C
    NEWREC(119:121)=E1OTH2C
    NEWREC(122:125)=E2TOTC
    NEWREC(126:128)=E2SPTC
    NEWREC(129:131)=E2SPNC
    NEWREC(132:134)=E2OTH1C
    NEWREC(135:137)=E2OTH2C
    NEWREC(138:141)=E3TOTC
    NEWREC(142:144)=E3SPTC
    NEWREC(145:147)=E3SPNC
    NEWREC(148:150)=E3OTH1C
    NEWREC(151:153)=E3OTH2C
    NEWREC(154:157)=KKILL
    NEWREC(158:160)=TGSPTK
    NEWREC(161:163)=TGSPNK
    NEWREC(164:166)=TGOTHK
    NEWREC(167:169)=TRSPTK

```

```

NEWREC(170:172)=TRSPNK
NEWREC(173:175)=TROTHK
NEWREC(176:178)=SKSPTK
NEWREC(179:181)=SKSPNK
NEWREC(182:184)=SKOTHK
NEWREC(185:187)=OTSPTK
NEWREC(188:190)=OTSPNK
NEWREC(191:193)=OTOTHK
NEWREC(194:196)=UKSPTK
NEWREC(197:199)=UKSPNK
NEWREC(200:202)=UKOTHK
NEWREC(203:206)=OBSTB
NEWREC(207:210)=OHIB
NEWREC(211:214)=OLOB
NEWREC(215:217)=OSPTB
NEWREC(218:220)=OESTB
NEWREC(221:223)=OWBB
NEWREC(224:226)=OSPN2B
NEWREC(227:229)=OOTH1B
NEWREC(230:232)=OOTH2B
NEWREC(233:236)=OBSTC
NEWREC(237:240)=OHIC
NEWREC(241:244)=OLOC
NEWREC(245:247)=OSPTC
NEWREC(248:250)=OESTC
NEWREC(251:253)=OWBC
NEWREC(254:256)=OSPN2C
NEWREC(257:259)=OOTH1C
NEWREC(260:262)=OOTH2C

```

C

C* WRITE OUT A RECORD TO THE OUTPUT FILE

C

```
WRITE(30,9007) NEWREC
```

```
9007 FORMAT(A262)
```

```
GO TO 50
```

C

C* IF NO MATCH ON THE SET LOG DATABASE, PRINT AN ERROR MESSAGE AND

C* READ IN THE NEXT LHAGE RECORD

C

```
990 PRINT *, 'NO MATCH ON SET LOG FOR SPECIMEN = ', LHAGEREC(1:7)
```

```
GO TO 50
```

C

C* AT THE END OF THE LHAGE DATAFILE, CLOSE THE INPUT AND OUTPUT FILES

C

```
995 CLOSE (UNIT=10)
```

```
CLOSE (UNIT=20)
```

```
CLOSE (UNIT=30)
```

```
STOP
```

```
END
```

APPENDIX 5D

```

C*****
C
C* PROGRAM:  OUTLIER
C
C* PURPOSE:  TO IDENTIFY SPECIMENS WITHIN A GIVEN RANGE OF
C             VALUES.  PROGRAM MUST BE EDITED TO SPECIFY INPUT
C             FILE, VARIABLES AND THEIR FORMAT, AND DESIRED
C             RANGES.  PROGRAM WILL PRINT SPECIMEN NUMBER AND
C             OTHER DESIRED OUTPUT TO THE TERMINAL.
C
C* PROGRAMMER:  G. JAY WALKER, FALL 1982, VAX FORTRAN
C
C*****
      IMPLICIT INTEGER (A-Z)
      CHARACTER SPEC*9
      OPEN (UNIT=10, NAME='FLHAGE81.DAT', STATUS='OLD',
1CARRIAGE CONTROL='LIST')
100  IVAL=0
      READ (10,9001,END=990) SPEC, TL, ACMLAVG, AAHLAVG, POOLM, CATOT
9001 FORMAT (A7,I4,2X,I5,11X,I5,9X,I5,7X,I2)
      IF((POOLM .LT. 1000).AND.(CATOT .GT. 0))IVAL=IVAL+100
      IF((AAHLAVG .LE. 800) .AND. ((ACMLAVG .GE. 1400)
+ .AND. (ACMLAVG .LT. 7000))) IVAL=IVAL+10
      IF (((AAHLAVG .GE. 800) .AND. (AAHLAVG .LT. 7000)).AND.
+ (ACMLAVG .LT. 500)) IVAL=IVAL+1
      IF (IVAL .NE. 0) THEN
        WRITE (6,9003) SPEC,IVAL
9003 FORMAT (2X,'SPECIMEN=',A7,3X,'IVAL=',I7)
      END IF
      GO TO 100
990  CLOSE (UNIT=10)
      STOP
      END

```


APPENDIX 6

Age Data File Formats

APPENDIX 6: Age Data File Formats

APPENDIX 6A

Age Data File: XXTRFSPT (original tooth reading record) and
FSPTLAST.DAT.

Format of XXTRFSPT

<u>Variable</u>	<u>Columns</u>
1. Reading of day	1:3
2. Specimen number	4:10
3. Tooth of specimen	11:11
4. Slide of tooth	12:12
5. Preparation date	13:18
6. Preparator	19:20
7. Tooth treatment	21:21
8. Mounting medium	22:22
9. Condition of slide	23:29
10. Reader code	30:31
11. Request remount/recut	32:32
12. Reading date	33:38
13. Microscope I.D.	39:40
14. Pulp cavity condition	41:41
15. Age recorded - Best	42:44
- High	45:47
- Low	48:50
16. Dentinal condition	51:53
17. Dentinal GLG's - Best	54:56
- High	57:59
- Low	60:62
18. Confidence	63:63
19. Cemental condition	64:66
20. Cemental GLG's - Best	67:69
- High	70:72
- Low	73:75
21. Confidence	76:76
22. Stock code	77:78
23. Sex	79:79
24. "Bad" reading	80:80

APPENDIX 6B

Age Data File: TRFSPTDB.DAT (reformatted and indexed)
 Program : REFORMAT
 Input File(s): XXTRFSPT.AOK

Format of TRFSPTDB		Columns
Variable		
1. Reader Code		1:2
2. Reading date		3:8
3. Reading of day		9:11
4. Specimen number		12:18
5. Tooth of specimen		19:19
6. Slide		20:20
7. Preparation date		21:26
8. Preparator		27:28
9. Tooth treatment		29:29
10. Mounting medium		30:30
11. Condition of slide		31:37
12. Request remount/recut		38:38
13. Remaining elements of tooth reading record in the same sequence		39:80

APPENDIX 6C

Age Data File: AGEBESTF
 Program: MAGEBST
 Input File(s): XXTRFSPT.AOK

Format of AGEBESTF		Columns
Variable		
1. Specimen number		2:8
2. Reader code 6		11:12
3. ACM mean		15:22
4. Standard deviation		25:32
5. ACM sample size		35:36
6. Reader code 18		40:41
7. AAH mean		44:51
8. Standard deviation		54:61
9. AAH sample size		64:66
10. Pooled mean age		69:76
11. Pooled variance estimate 1		82:86
12. Pooled variance estimate 2		92:96
13. T value		102:106
14. Significance		109:114

APPENDIX 6D

Age Data File: FLHAGE
 Program : MAKEDAT
 Input Files : FSPTDB.DAT
 AGEBESTF.DAT
 FSPTLAST.DAT

Format of FLHAGE

<u>Variable</u>	<u>Columns</u>
1. All life history elements	1:151
2. Pooled mean age	152:156
3. ACM mean	157:161
4. Age recorded- Best	162:164
- High	165:167
- Low	168:170
5. AAH mean	171:175
6. Age recorded- Best	176:178
- High	179:181
- Low	182:184

APPENDIX 6E

Age Data File: FSETAGE
 Program : MAKEDAT2
 Input File(s): FLHAGE.DAT
 SLyrDB.DAT

Format of FSETAGE

<u>Variable</u>	<u>Columns</u>
1. Specimen number	1:7
2. Total length	8:11
3. Pooled mean age	12:16
4. ACM Last Age Recorded - Best	17:19
- High	20:22
- Low	23:25
5. AAH Last Age Recorded - Best	26:28
- High	29:31
- Low	32:34
6. Selected Set Log Variables	35:262

(See MAKEDAT2 program for output record.)

